PROYECTO FIN DE GRADO

SIMULACIÓN DE LA AUTOMATIZACIÓN DE PROCESOS CON UNITY PRO

AUTOR:

Guillermo Calvo Guadaño

INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TUTOR/A: DOLORES BLANCO ROJAS

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

ESCUELA POLITÉCNICA

UNIVERSIDAD CARLOS III DE MADRID

JULIO 2013

AGRADECIMIENTOS

Quería agradecer a toda mi familia por todos estos años, en especial a mi madre que siempre ha estado ahí dando esos empujones finales cuando más lo he necesitado, simplemente gracias Moli. También agradecer a Dolores por su ayuda y colaboración.

INDICE DE CONTENIDOS

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

ÍNDICE GENERAL

INDICE DE FIGURAS

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. MOTIVACION

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

1 MOTIVACIÓN

1.1 Motivación del Proyecto

El proyecto surge de la creación del nuevo software Unity Pro XL para los autómatas Premium de Schneider del laboratorio de Automatización del departamento de Ingeniería de Sistemas y Automática. Debido a mi afición por la programación de los PLC´s y la puesta en marcha del mismo, así como los sistemas SCADA y la comunicación entre el autómata y los procesos correspondientes, me decliné por el proyecto. El funcionamiento de este software es muy similar a la de su predecesor, el PL7 Junior, pero tiene una nueva herramienta muy útil a la hora de trabajar sin el autómata. Esta herramienta es el simulador, herramienta muy importante para el trabajo fin de grado. Este documento, o parte de él, pretende ser utilizado como manual de uso del simulador, ya que los alumnos no siempre tienen disponible el acceso a los laboratorios. El uso del simulador permite que el alumno pueda trabajar fuera de los laboratorios, en su casa, o en aulas informáticas, lo que facilita el estudio del manejo de los autómatas.

1.2 Objetivos

El objetivo de este trabajo fin de grado es el desarrollar un documento que sirva, como manual de uso del simulador del software Unity Pro a nivel de usuario y también se pueda aplicar a la docencia, de tal forma que el alumno, con unas pautas establecidas, pueda programar a su gusto la estación creada. Este documento debe aportar toda la información necesaria para realizar una simulación completa de un proceso automatizado, desde la creación del proyecto, la programación, el diseño gráfico del sistema, hasta su simulación para observar el avance de los distintos estados del mismo. Dicho manual, irá orientado a todos los alumnos estudiantes de la asignatura de Automatización Industrial, y de todas las asignaturas en las que se utilice el PLC o autómata.

Para ello se deberán cumplir los siguientes sub-objetivos:

- Familiarizarse con el simulador y probar su correcta conexión y funcionamiento.
- Orientar el documento a un lenguaje docente de manera que los conceptos utilizados y los pasos a realizar queden completamente claros para los alumnos.
- Desarrollar la solución de programación y las pantallas de operador para un ejemplo de automatización propuesto, analizando las posibilidades que el software UnityPro ofrece.

1.3 Estructura del documento

El proyecto está estructurado de la siguiente manera:

- El capítulo de la "Introducción", que es el numerado como Capítulo 2, explica por qué la automatización es importante en el ámbito docente, además de presentar los autómatas y los medios de los que dispone la Universidad Carlos III para dar a conocer a sus alumnos este referente. Se define lo que es un simulador, sus funciones y aplicaciones. Para completar este capítulo, en él se presta atención al uso de los simuladores en la docencia y las ventajas que aportan.

- El capítulo siguiente, el Capítulo 3, es el llamado "Sotfware Unity Pro". En él se realiza un recorrido por el funcionamiento general del software. Se presenta una descripción del software y de sus características principales. Se hace un repaso de la estandarización de la programación del control industrial y del cumplimiento de la norma IEC 61131. A partir de esta norma se hace distinción de la programación que se considera lenguaje, tanto literal como gráfico, y de lo que simplemente es considerado una estructura organizada. Esta aclaración es de gran importancia, porque no siempre ha sido igual debido a los cambios y fusiones que han surgido en los lenguajes de programación en la automatización industrial desde su nacimiento. El capítulo continúa con la presentación de las variables, direcciones y bits de palabras y sistemas disponibles en Unity Pro. A continuación, se describe el simulador y se detallan las opciones que ofrece tratar con un simulador. Para finalizar, se aclara la diferencia existente entre un sistema Scada y un simulador.

- El Capítulo 4 se llama "Ejemplo desarrollado", donde se ha elegido un problema tipo de la automatización de un proceso, y se ha programado por completo con la herramienta de estudio. Tras la presentación del problema y la configuración del entorno de trabajo, se explica cómo se programa el ejemplo con dos soluciones diferentes.

- El Capítulo 5, "Simulación del ejemplo programado" se detalla la realización de una "pantalla del operador". Todas las posibilidades de representación que existe en la librería y modo de programar cada tipo de elemento. Se expone un desarrollo de una simulación completa. Además de explicar cómo se realiza la compilación, la transferencia y el desarrollo del proyecto con el simulador de Unity Pro.

- En el Capítulo 6 se presentan las conclusiones y además posibles trabajos futuros relacionados con el tema de este proyecto.

2. INTRODUCCION

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

2. INTRODUCCIÓN:

En este capítulo, veremos la parte de la automatización en la docencia, donde el objetivo mas claro es proporcionar al alumno un conocimiento más alto de los equipos y de los sistemas de automatización, así como la función del simulador y las ventajas que éste tiene con la parte docente.

2.1 La automatización y su importancia en la docencia:

La definición de Automatización Industrial (automatización; del griego antiguo *auto*: guiado por uno mismo) es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales sustituyendo a operadores humanos. [1]

La automatización como una disciplina de la ingeniería que es más amplia que un mero sistema de control, abarca la instrumentación industrial, que incluye los sensores, los sistemas de control y supervisión, los sistemas de transmisión y toma de datos y las aplicaciones de software en tiempo real para supervisar, controlar las operaciones de plantas o procesos industriales. [2]

En cuanto al concepto de uno de los elementos principales que es el Autómata programable industrial (PLC), es una computadora utilizada en ingeniería automática o automatización industrial, diseñado para controlar procesos secuenciales (una etapa después de la otra) que se ejecutan en un ambiente industrial. Es decir, que van asociados a la maquinaria que desarrolla procesos de producción y controlan su trabajo.

También puede definirse como equipo electrónico de control con un cableado interno, denominado hardware, independiente del proceso a controlar, que se adapta a éste mediante un programa específico, software.

El PLC en sí, es un sistema, porque contiene todo lo necesario para operar, y es industrial, por tener todos los registros necesarios para operar en los ambientes hostiles que se encuentran en la industria.

Con las señales de entrada se puede controlar tanto elementos digitales, como finales de carrera y detectores de proximidad. También pueden ser analógicos, como sensores de temperatura o sensores de presión, así como dispositivos de salida en tensión o corriente. De la misma manera, las señales de salida pueden ser digitales, todo o nada, o señales analógicas, por ejemplo de tensión o corriente, que sirven a los elementos indicadores y actuadores del proceso.

El PLC gobierna las señales de salida según el programa de control previamente almacenado en una memoria, a partir del estado de las señales de entrada y del estado actual del proceso. Este programa se introduce en el autómata a través de la unidad de programación, que permite además funciones adicionales como depuración de programas, simulación, monitorización, control del autómata.

La principal característica que diferencia el PLC frente a otros sistemas de control programables es la estandarización de su hardware, que permite la configuración de sistemas

de control adaptados a las necesidades estimadas de potencia de cálculo, y tipo de señales de entrada y salida.

En instalaciones en donde es necesario un proceso de maniobra, control, señalización, su utilización es fundamental para un correcto funcionamiento. Su aplicación abarca desde procesos de fabricación de cualquier tipo a transformaciones industriales hasta control de instalaciones.

El objetivo de las aulas docentes de la Universidad Carlos III, es orientar al alumno hacia el correcto control y desarrollo de proyectos de automatización industrial, donde los PLC, los sistemas SCADA así como los sistemas neumáticos forman parte de dichos proyectos.

El autómata programable utilizado para la realización del proyecto ha sido el Premium TSX de la compañía Schneider, dicho PLC lo forman estos elementos:

- Fuente de alimentación
- Bastidor
- Procesador
- Módulo de entradas digitales (DI)
- Módulo de entradas analógicas (AI)
- Módulo de salidas digitales (DO)
- Módulo de salidas analógicas (AO)
- Cable de comunicación con PC
- Telefast (Simplificación de cableado)

La configuración del PLC dependerá del número de tarjetas que se requiera para el desarrollo del proyecto que se desee. La imagen del autómata que utilizamos es la que aparece en la figura. [3]

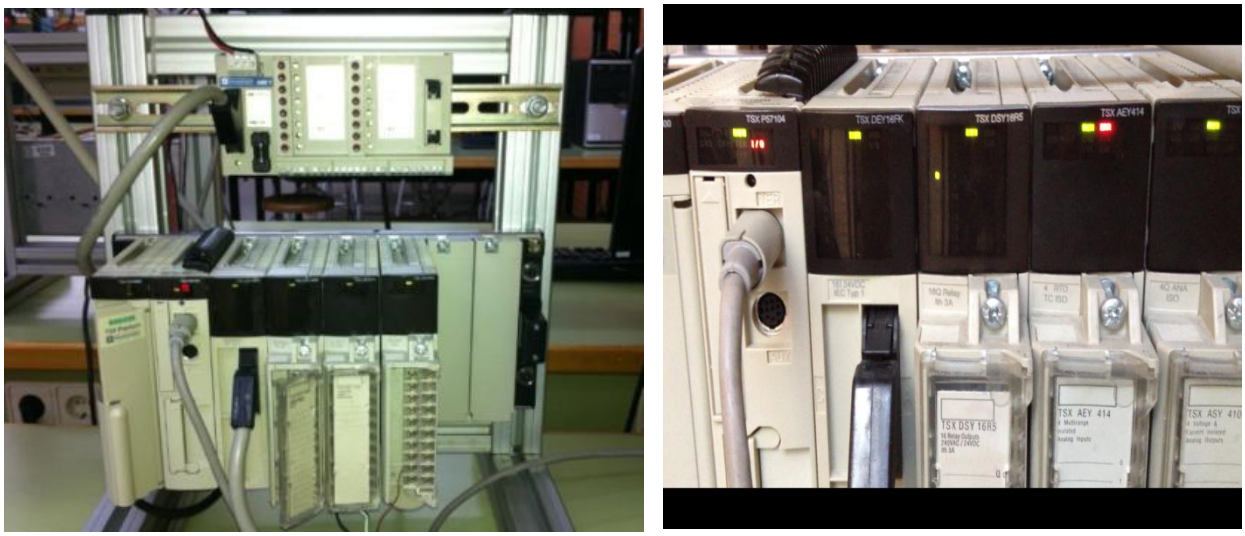


Figura 1: Autómata de laboratorio UC3M

2.2 El Simulador y su uso

La introducción de simuladores ha sido muy significativa en los programas de formación en campos como la automoción, la industria militar y la aviación.

Un simulador es un aparato, por lo general informático, que permite la reproducción de un sistema. Los simuladores reproducen sensaciones y experiencias que en la realidad pueden llegar a suceder.

Un simulador pretende reproducir tanto las sensaciones físicas (velocidad, aceleración, percepción del entorno) como el comportamiento de los equipos de la máquina que se pretende simular. Para simular las sensaciones físicas se puede recurrir a complejos mecanismos hidráulicos comandados por potentes ordenadores que mediante modelos matemáticos consiguen reproducir sensaciones de velocidad y aceleración. Para reproducir el entorno exterior se emplean proyecciones de bases de datos de terreno. [4]

Entre los simuladores más destacados:

- **Simulador de conducción**: permiten a los alumnos de autoescuela, enfrentarse con mayor seguridad a las primeras clases prácticas.
- **Simulador de carreras**, con los que se puede conducir un automóvil, motocicleta, camión, etc.
- **Simulador de vuelo o de aviones**: permite dominar el mundo de la aviación y pilotar aviones, helicópteros...
- **Simulador de trenes**: permite controlar un tren.
- **Simulador de negocio**: permite simular un entorno empresarial.
- **Simulador clínico médico**: permite realizar diagnósticos clínicos sobre pacientes virtuales.
- **Simulador musical**: permite reproducir sonidos con un instrumento de juguete.
- **Simulador termo-solar**: permite analizar la influencia de la producción de electricidad en la modificación de ciertos parámetros en una central solar termoeléctrica.

2.3 El simulador y su aplicación a la parte docente

Los simuladores educativos son programas que permiten hacer prácticas simuladas, pero con las mismas características que en la realidad. Permiten ahorrar costes y reducir riesgos pero manteniendo la calidad en la formación.

La simulación y el aprendizaje son dos conceptos que están ligados en el proceso educativo. Podemos decir que la mayoría de las actividades de aprendizaje siempre están basadas en entidades de simulación. En todo momento profesor y alumno están trabajando con hipótesis y supuestos, ya que no se pueden acceder al exterior para explicar y demostrar los procesos establecidos.

Simplifica las operaciones con la integración de gestión y producción. La utilización de un simulador en el marco del constructivismo en la educación, para promover el desarrollo de habilidades y actitudes con los alumnos en clase, permite ampliar las técnicas y herramientas tecnológicas para un mayor aprendizaje. [5]

Entre los retos a resolver en este siglo en la educación, destaca el gran reto de desarrollo e implementación de estrategias de enseñanza. Conseguir métodos dirigidos a desarrollar habilidades y actitudes en los estudiantes, para que adquieran distinciones y competencias. Éstas permiten estar funcionalmente activos en lo profesional y ser capaces de tomar decisiones que les lleven a resolver genuinamente los problemas. Además, demanda del alumno el ejercicio de compartir los recursos y conocimientos de que dispongan, a través de la práctica de aprendizajes colaborativos.

La función de los simuladores está basada en aprendizaje de tipo experimental, y en conjeturas, para llevar a cabo un aprendizaje basado en el descubrimiento. El alumno obtiene conocimiento por medio de la interacción con un mundo simulado logrando así controlar una situación del mundo real, mediante el cual se logrará observar diferentes situaciones, y aprenderá a tomar las decisiones.

2.4 Ventajas de los simuladores: simulación

La simulación como herramienta de apoyo al estudio presenta numerosas ventajas, si bien es cierto que, como instrumento que es, debe ser bien utilizada.

Entre las principales ventajas que nos permiten los simuladores:

- El **Aprendizaje** por Descubrimiento: Es una forma activa de aprender en la que el alumno es el propio artífice de su aprendizaje. Se sugieren al alumno unas hipótesis y éste las desarrolla buscando las causas y efectos de los distintos fenómenos. Básicamente se trata de que el alumno sea capaz de analizar sistemáticamente los fenómenos y probar el comportamiento de un modelo en distintos escenarios.
- Fomentar la **creatividad**: Es una de las ventajas de los entornos de simulación. La posibilidad de disponer de cajas de herramientas y colecciones de bloques-operadores en los entornos permite la disponibilidad de un laboratorio, taller, o mesa de diseño con la que el alumno pueda no sólo simular modelos que se le den hechos sino que pueda construir los suyos propios. En esta parcela el diseño de máquinas es ideal. En este sentido los entornos de simulación han de ser flexibles y multifuncionales.
- **Ahorrar** tiempo y dinero: Ninguna de las dos cuestiones es banal en la actualidad educativa de nuestro mundo. Procesar la información no es tarea fácil, y la adquisición, ordenación, tratamiento y análisis de la información son aspectos muy importantes de cara al proceso de aprendizaje. Ya han perdido sentido aquellas teorías de aprendizaje en las que el alumno, por repetición oral o escrita aprendía las lecciones. La cantidad de conocimientos que hay que aprender hace necesario el utilizar técnicas de aprendizaje que aceleren el proceso. La simulación es una de ellas. Sin descartar los procesos constructivos y manipulativos del aprendizaje. El ordenador es capaz de trabajar por nosotros evitándonos los procesos repetitivos de cálculo. El ahorro que reporta el uso de estas herramientas de simulación es evidente ya que sustituimos los equipos de entrenamiento, laboratorios y plantas de ensayo por un entorno virtual.
- La **enseñanza individualizada**: Las herramientas de simulación permiten que el alumno lleve su propio ritmo de aprendizaje y se enfrente de modo individual al proceso de elaboración de sus propias conclusiones con relación a los fenómenos que va a simular. Algunos entornos de simulación prevén el aprendizaje individual realizando una tutorización guiada del aprendizaje de tal manera que incorporan bases de

conocimiento en las que el profesor modela el proceso de aprendizaje mediante planes de estudio.

- La **autoevaluación**: La simulación permite al alumno realizar acciones orientadas a su propia autoevaluación mediante el planteamiento de guiones y cuestionarios orientados al tema que está estudiando. [6]

3. SOFTWARE UNITY PRO XL SCHNEIDER

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

3. SOFTWARE UNITY PRO XL- SCHNEIDER:

Unity Pro es un software común de programación, puesta a punto y explotación de los autómatas Modicom M340, Premium y Quantum, estudiaremos el caso de la versión 5.0 del software.
El software se adhiere a la normativa IEC 61131-3, Unity Pro surge de la experiencia en el software PL7 y Concept. Además abre las puertas de un conjunto completo. También se explicarán las posibilidades que ofrece la utilización de las variables, direcciones y bits de palabras.

3.1 Descripción y características del software

La característica principal de este software Unity Pro XL, es que es de última generación y está implantado muy recientemente. Más adelante se verán todas las operaciones que contiene y la versatilidad que presenta.

Dentro del software Unity Pro XL se puede generar proyectos en varias familias de autómatas, como son Modicom M340, Premium, Quantum y Atrium. Todos ellos dentro de la marca Schneider Electric.

El software industrial está basado en formatos estándares, reduciendo sensiblemente las actividades de familiarización y capacitación. Además, presenta las herramientas necesarias para la creación, depuración y puesta en marcha de las aplicaciones. El entorno de ejecución de los programas es Windows 98/2000/NT/XP/7, estando adaptado a su funcionamiento gráfico y orientado a objetos.

Es Software "todo en uno" y utilización sencilla y aprovecha al máximo las ventajas de los interfaces gráficos y contextuales de Windows 98/2000/NT/XP/7:

- Acceso directo a las herramientas y a los datos.
- Configuración 100% gráfica
- Barra de herramientas e iconos personalizables.
- Funciones avanzadas de "arrastrar y soltar" y zoom.
- Ventana de diagnóstico integrado.

Las ventajas de la estandarización en Unity Pro proponen un conjunto completo de funcionalidades y de herramientas que permiten calcar la estructura de la aplicación en la estructura del proceso o de la máquina. El programa se divide en módulos funcionales jerarquizados que agrupan:

• Secciones de programa.
• Tablas de animación.
• Pantallas de los operadores.
• Hipervínculos.

Las funciones básicas, utilizadas de forma repetitiva, se pueden integrar en bloques de funciones de usuario (DFB) en lenguaje IEC 61131-3.

3.2 Estandarización en la programación de control industrial

Actualmente aún siguen persistiendo sistemas de control específicos del fabricante, con programación dependiente y conexión compleja entre distintos sistemas de control. Esto significa para el usuario costos elevados, escasa flexibilidad y falta de normalización en las soluciones al control industrial.

En cuanto a la normativa IEC 61131-3:

- Es la base real para estandarizar los lenguajes de programación en la automatización industrial, haciendo el trabajo independiente de cualquier compañía.
- Es el resultado del gran esfuerzo realizado por multinacionales a los que se añaden muchos años de experiencia en el campo de la automatización industrial.
- Las especificaciones de la sintaxis y semántica de un lenguaje de programación, incluyendo el modelo de software y la estructura del lenguaje, es una de las maneras de escribir el trabajo desarrollado.

Bajo esta norma, se definen cuatro lenguajes de programación normalizados. Esto significa que su sintaxis y semántica ha sido definida, no permitiendo particularidades distintivas (dialectos).

Los lenguajes consisten en dos de tipo literal y dos de tipo gráfico:

Literales: * Lista de instrucciones (Instruction List, **IL**).
* Texto estructurado (Structured Text, **ST)**.

Gráficos: * Diagrama de contactos (Diagram Ladder, **LD**).
* Diagrama de bloques funcionales (Function Block Diagram, **FBD**).

En la figura 2, los cuatro programas describen la misma acción. La elección del lenguaje de programación depende de:

- Los conocimientos del programador.
- El problema a tratar.
- El nivel de descripción del proceso.
- La estructura del sistema de control.
- La coordinación con otras personas o departamentos.

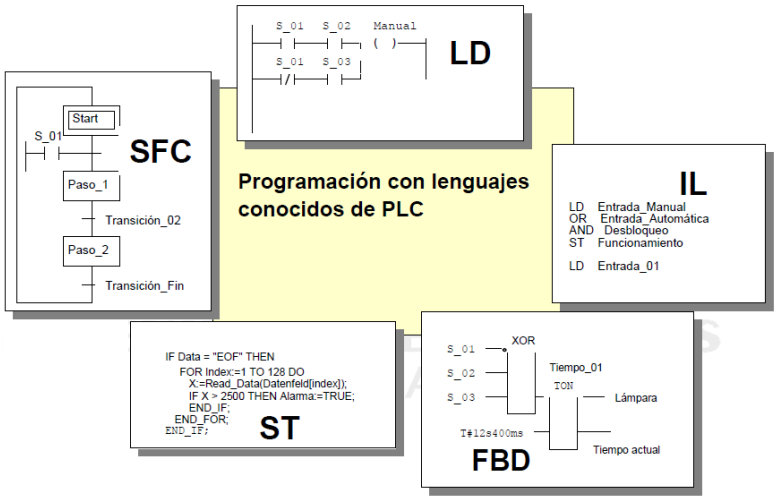


Figura 2: Lenguajes IEC-1131-3

Los cuatros lenguajes están interrelacionados y permiten su empleo para resolver conjuntamente un problema común según la experiencia del usuario.

El Diagrama de contactos **(LD)** tiene sus orígenes en los Estados Unidos. Está basado en la presentación gráfica de la lógica de relés.

Lista de Instrucciones **(IL)** es el modelo de lenguaje ensamblador basado un acumulador simple; procede del alemán ´Anweisungliste, AWL.

El *Diagramas de Bloques Funcionales* (FBD) es muy común en aplicaciones que implican flujo de información o datos entre componentes de control. Las funciones y bloques funcionales aparecen como circuitos integrados y es ampliamente utilizado en Europa.

El lenguaje *Texto estructurado* (ST) es un lenguaje de alto nivel con orígenes en el Ada, Pascal y ´C´; puede ser utilizado para codificar expresiones complejas e instrucciones anidadas.

La norma IEC 61131-3 también permite dos formas de desarrollar tu programa de control (ver figura 2): de arriba a abajo (Top-down) y de abajo a arriba (bottom-up). Puedes especificar inicialmente la aplicación completa y dividirla en partes, declarar las variables y demás. También puedes comenzar la programación desde abajo, por ejemplo, por medio de funciones y bloque funcionales. Por cualquiera de los caminos que elijas, IEC-1131-3 te ayudará durante todo el proceso.

Figura 3: Norma IEC 61131-3

Cumplir todos los requerimientos de la norma IEC 1131-3 no es fácil, por eso se permiten implementaciones parciales en varios aspectos. Esto hace referencia al número de lenguajes que soportan las herramientas de desarrollo disponibles, y al número de funciones y de bloques funcionales. Con ello se deja libertad al suministrador, pero el usuario debe tener cuidado durante el proceso de selección de la herramienta adecuada. Incluso una actualización del software puede dar lugar a un nivel muy alto de trabajo durante la implementación.

3.3 Tipos de variables, direccionamiento, bits y palabras del sistema

3.3.1 Variables

Los tipos de variables utilizadas para la realización del proyecto se tienen que definir en el software de Unity Pro, con sus respectivas entradas, salidas, temporizadores, contadores, etc.
Los tipos de variables que se utilizan, bien para operar o bien comandos de entradas para temporizadores, contadores son variables de tipo:

- INT: Sigla del formato single INTEGER (entero simple) (codificado en 26 bits). Los límites inferior y superior figuran a continuación: de –2^15 a 2^15.

- BOOL: Sigla del tipo Booleano. Se trata del elemento de datos de base informática. Una variable de tipo BOOL posee uno de estos valores: 0 (FALSE) o 1 (TRUE).

- EBOOL: Sigla del tipo Extended BOOLEAN (booleano extendido). Una variable de tipo EBOOL posee el valor 0 (FALSE) o 1 (TRUE), pero igualmente los flancos ascendentes o descendentes y las funciones de forzado. Una variable de tipo EBOOL ocupa un byte de memoria.

- TIMER: Sigla del tipo temporizador. Es una variable de tiempo en la que el valor esperado es el valor en segundos. Se utiliza normalmente para temporizadores.

- STRING: una cadena de caracteres, palabra, ristra de caracteres o frase (*string* en inglés) es una secuencia ordenada de longitud arbitraria (aunque finita) de elementos que pertenecen a un cierto lenguaje formal.

- WORD: Representa una cadena de 16 bits, significando una longitud de datos de 16 bits.

3.3.2 Direccionamiento

Existen 2 tipos de direccionamientos en nuestro PLCs que serían “topológicos” y de “memoria”.
Topológica: Nos indica la dirección física donde estarían nuestro hardware, puertos comunicaciones, pcmcia, tarjetas de expansión, etc.
Memoria: Este tipo de direccionamiento lo utilizamos para leer/escribir en unas ciertas zonas de memoria como pueden ser E/S, bits, palabra, doble palabra…en la llamada sintaxis IEC61131.

- **%I**: Indica un objeto de lenguaje de entrada binario.
- **%W**: Indica un objeto de lenguaje de entrada analógico.
- **%KW**: Indica un objeto de lenguaje de palabra constante.
- **%M**: Indica un objeto de lenguaje de bit de memoria.
- **%MW**: Indica un objeto de lenguaje de palabra de memoria.
- **%Q**: Indica un objeto de lenguaje de salida binaria.
- **%QW**: Indica un objeto de lenguaje de salida analógica.

3.3.3 Bits y palabras del sistema

Existen unos bits y palabras del sistema que son de gran utilidad e importancia a la hora tanto de programar como de depurar la programación en busca de errores y estados del autómata. Los autómatas Modicon M340, Premium, Atrium y Quantum utilizan bits de sistema *%S* que indican los estados del autómata o que permiten controlar el funcionamiento de éste. Dichos bits pueden probarse en el programa del usuario con el fin de detectar cualquier evolución de funcionamiento que conlleve un procedimiento de procesamiento establecido. Algunos de estos bits deben volver a su estado inicial o normal por programa.
Entre los bits más destacados están:

- **%S0**, está normalmente en 0. Este bit se define a 1 durante el primer ciclo completo de restauración del PLC en modalidad RUN o STOP, pero no en caso de modo simulación.
- **%S4**, es un bit que se encuentra un tiempo de simulación encendido y otro apagado, con este bit simulamos el parpadeo de la luz de emergencia.

3.4 El simulador de Unity Pro XL en comparativa con el PLC

Una de las herramientas más importantes a la hora de realizar el proyecto con el software de Unity Pro ha sido el entorno de simulación que posee, que nos permite representar de manera muy precisa usando pantallas de operador el proceso que estamos programando. Sin duda una gran comodidad para el programador, ya que puede comprobar el estado del proceso desde su propio ordenador, sin tener que desplazarse hasta el PLC.

El simulador nos ofrece la posibilidad de cambiar el estado de las entradas de OFF a ON y viceversa con solo actuar el ratón, también nos permite visualizar el estado de las salidas mediante indicadores luminosos.

3.4.1 El simulador frente al PLC

Existen algunas diferencias entre el simulador y el PLC, a continuación se describen algunas de las más importantes:

- El simulador no admite ninguna forma de Entrada/Salida (E/S). Aunque la simulación contiene los componentes del proyecto para E/S, el simulador no los procesa. A las entradas y salidas sólo se puede acceder desde el proyecto o a través de las funciones online de Unity Pro (leer, escribir, forzar, animar, etc.).

- El comportamiento de tiempo de ejecución del simulador no es equiparable al comportamiento de tiempo de ejecución de un PLC real, es mucho más rápida la ejecución en el PLC real.

- El simulador no soporta algunos bits de sistema, como por ejemplo, el bit de arranque en frío %S0 que durante el primer ciclo completo de restauración del PLC en modalidad RUN o STOP se mantiene activado, en el simulador se mantiene a cero (desactivado).

3.5 Diferencias entre sistemas SCADA y el simulador

SCADA, acrónimo de Supervisory Control And Data Acquisition (Supervisión, Control y Adquisición de Datos) es un software para ordenadores que permite controlar y supervisar procesos industriales a distancia. Facilita retroalimentación en tiempo real con los dispositivos de campo (sensores y actuadores) y controlando el proceso automáticamente. Provee de toda la información que se genera en el proceso productivo (supervisión, control calidad, control de producción, almacenamiento de datos, etc.) y permite su gestión e intervención.[8]

Existe diferencia entre el SCADA y el simulador, y es que en el simulador se simulan las variables configuradas y durante la simulación, es posible manipular, activar y desactivar dichas variables. En el SCADA se puede supervisar las tareas de otros individuos, aunque esto último no significa que tenga el control sobre ellos, sino que el objetivo principal es correctivo. Para que la modificación sea posible, el sistema SCADA puede tener realimentación.

Una de las maneras de integrar un sistema SCADA en Unity Pro, es comunicando un sistema de control mediante redes o buses, el SCADA de Schneider como tal se llama Vijeo Citect. Otra posibilidad es mediante el Vijeo Designer Runtime. [9]

Si quisiéramos unir el PLC con el SCADA, sería necesaria previamente configurar el HW necesario que realizará la configuración de la red Ethernet a la que estará conectada el PC o PLC. Por lo tanto habría que realizar la definición de la red Ethernet, así como su asignación al puerto Ethernet de la CPU.

4. EMJEMPLO DESARROLLADO: TÚNEL AUTOLAVADO

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

4 EJEMPLO DESARROLLADO: TÚNEL AUTOLAVADO

En este capítulo mostraremos un problema práctico de cómo simular la automatización de un sistema físico propuesto. En este ejemplo, se parte de un diagrama SFC, y dentro de cada etapa de este diagrama está programada con un tipo de lenguaje distinto, para comprobar la gran versatilidad que tiene el software.

Para conseguir un diagrama gráfico SFC, se programará las etapas y transiciones. Si la complicación de una etapa es elevada, se recurre a una herramienta disponible en Unity Pro denominada macro etapa. Los lenguajes utilizados han sido ST, IL, SFC, LD y FBD.

4.1 Presentación del problema

Se trata de automatizar un túnel de autolavado, desde que se introduce la moneda para que se abra la barrera, pasando por las etapas de prelavado, rodillos, aclarado y secado, hasta la salida del coche del túnel. Todo el proceso cuenta con sus correspondientes setas de emergencias e indicadores de funcionamiento y mal funcionamiento, así como un contador de rearmes, la distancia recorrida por el vehículo, control de la cinta transportadora que guía al vehículo para que no desvíe la trayectoria en ningún momento y el control de los tanques de agua, cera y detergente. En todo momento debe controlarse todos los elementos importantes para su funcionamiento y para ello se instalará un sistema de alarmas que avisará al usuario de cualquier imperfección. La instalación completa se representa en la Figura 4:

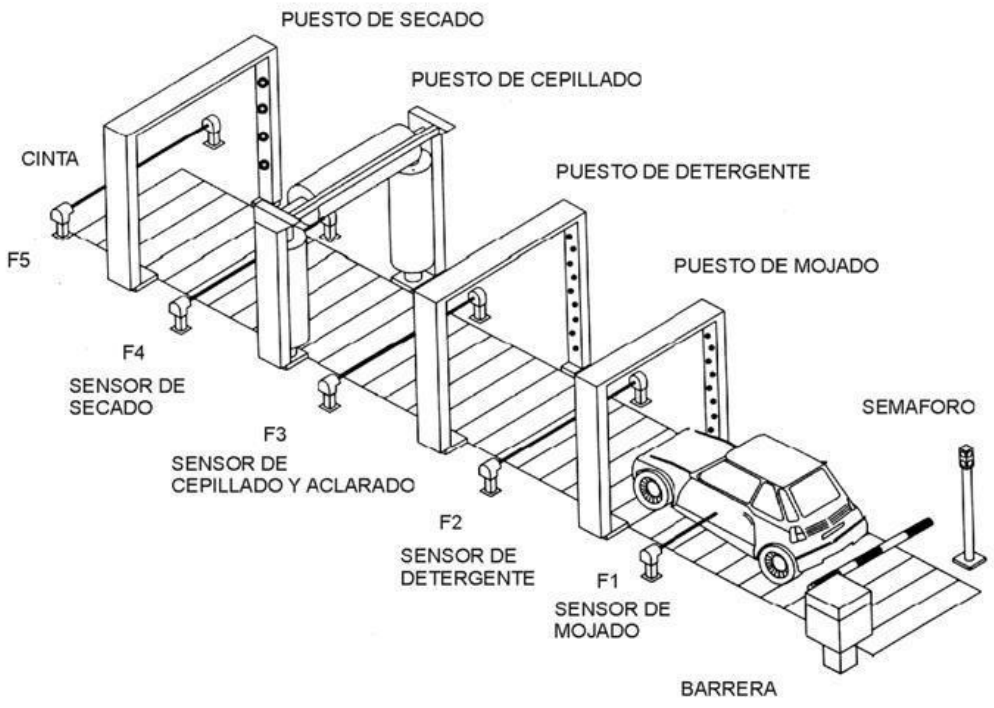


Figura 4: Instalación completa

El funcionamiento será de la siguiente manera:

Condiciones de funcionamiento

• El sistema de lavado consta de tres programas de funcionamiento, cada programa con sus respectivas características que vienen en el tablón de la estación.

• El sistema de lavado estará disponible siempre que se haya elegido el programa adecuado y a continuación se presione el Pulsador "Puesta en servicio"

• Que no esté pulsada ninguna seta de emergencia ("Seta de emergencia 1" y "Seta de emergencia 2 ") y que el sistema no haya sufrido más de 3 paradas de emergencia / problemas en cuyo caso para el rearme se debe hacer con una llave especial ("Llave Reset rearmes").

• El número de rearmes se contabilizará con el contador "Contador de rearmes" que será reseteado como la luz de emergencia de nuevo a valor 0 cuando se active la "Llave Reset Rearmes".

• Las setas de emergencia serán contactos normalmente cerrados. Si el sistema está dispuesto para funcionar se indicará mediante una lámpara de señalización de "Luz de servicio disponible".

• Si el sistema sufre alguna parada de emergencia se indicará por medio de la "Luz de emergencia". Esta luz debe parpadear con la frecuencia indicada=marca de ciclo. La parada de emergencia solo se considera como tal si el sistema está en marcha y debe desactivar todas las salidas. Una vez listo para el funcionamiento el sistema, el vehículo realizará las siguientes etapas:

1º.- Proceso de inicio y "Prelavado y Detergente"

• Cuando el sensor "**Detector1 Lavado Inicial**" D1 detecte el coche, entrará en funcionamiento la Primera fase del lavado que es la de "**Prelavado y Detergente**". Para ello se activarán los "Rodillos de avance" que permitirán arrastrar al coche durante todo el trayecto por el tunel.

• Simultáneamente también se actuará (valor 1) sobre la "**Válvula agua prelavado**" para comenzar a mojar el coche. Este proceso deberá durar 8 segundos controlados por el "**Temporizador Agua Prelavado**" T1.

• Transcurridos los 8 segundos, se cerrará la "**Válvula Agua Prelavado**" y se actuará sobre la "**Válvula agua detergente**" para proceder a liberar detergente sobre el coche. Este proceso durará 12 segundos y estará controlado por el "**Temporizador Agua Detergente**" T2, cerrándose en dicho momento dicha válvula.

2º.- Proceso de Rodillos y Limpieza

• Cuando el coche pase por el "**Detector2 Rodillos de Limpieza**" D2 se acabará el proceso anterior y comenzará el Proceso de Rodillos y Limpieza y se activarán (Valor 1) los "Rodillos de Limpieza" y los "Rodillos limpiallantas".

• Este estado se mantendrá hasta que el coche sea detectado por el "Detector3 aclarado" D3.

• En ese momento se desactivarán (valor 0) los "Rodillos de Limpieza" y los "Rodillos limpiallantas".

3ª.- Proceso de Aclarado y Abrillantado.

• Al entrar en esta fase, se actuará (valor 1) sobre la "Válvula agua aclarado" para iniciar el Proceso de Aclarado del coche.

• Este proceso durará 7 segundos controlados por el "Temporizador Agua Aclarado" T3.

• Transcurridos los 7 segundos se cerrará dicha válvula (valor 0) y se actuará sobre la "Válvula agua cera" para proceder a liberar cera abrillantadora sobre el coche. Este proceso deberá durar 15 segundos y estará controlado por el "Temporizador agua cera" T4 cerrándose en dicho momento dicha válvula.

• Cuando el coche sea detectado por el "Detector 4 secado" D4 se acabará el proceso anterior iniciándose la fase del Secado.

4º.- Proceso de Secado

• En ese momento se actuará sobre los motores de las "Soplantes de Secado"

• Estas deberán funcionar primero a una "Velocidad lenta soplante" durante 5 segundos controlados por el temporizador T5 "Temporiz veloc lenta".

• Transcurridos los 5 segundos pasarán a funcionar a una "Velocidad rápida soplante" durante 15 segundos controlados por el temporizador T6 "Temporiz veloc rápida".

• Al finalizar este tiempo se pararan (valor 0) los motores de las "Soplantes de Secado".

• En cuanto a las velocidades lenta y rápida de las soplantes, el valor de velocidad lenta será 1500 rpm que será cargado en la palabra de marcas MW6 "Velocidad lenta soplante" y el valor de la velocidad rápida será 3000 rpm que será cargado en la palabra de marcas MW8 "Velocidad rápida soplante".

5º.- Proceso de Salida

• Por último, cuando se detecte el coche por el "Detector5 final" D5 el coche habrá salido del túnel de autolavado permitiéndose la entrada de un nuevo coche para que se repita todo el proceso.

Si en cualquier momento durante el funcionamiento se pulsa alguna seta de emergencia, se cerrarán todas las válvulas y se pararan todos los motores.

4.2 Configuración del entorno de trabajo y definición de variables necesarias

Una vez tengamos planteado el problema hay que especificar el tipo de modelo de autómata que vamos a utilizar y con el que vamos a trabajar.
Dentro de las tres familias de PLC's que posee la gama de Schneider Electric, se va a seleccionar la Premium. Hemos elegido porque es la disponible en el laboratorio de Automatización Industrial de la Universidad Carlos III.
Abrimos Unity Pro, se crea un nuevo proyecto. La manera de crear un nuevo proyecto es en el menú Fichero, seleccionar "*Nuevo*". Aparecerá un cuadro como el que se muestra en la Figura 5. En él se elegirá la familia seleccionada, el "*modelo TSX P57 104M 0260 57-1":*

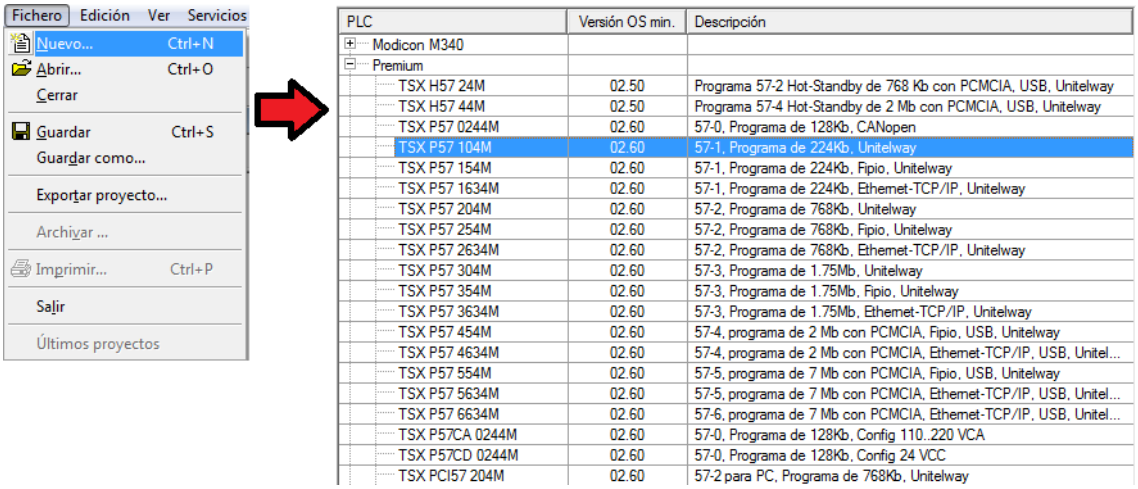


Figura 5: Elección de la CPU en Unity Pro

Una vez hemos configurado el entorno, se nos genera automáticamente el entorno de trabajo donde vamos a trabajar y realizar todo nuestro proceso.

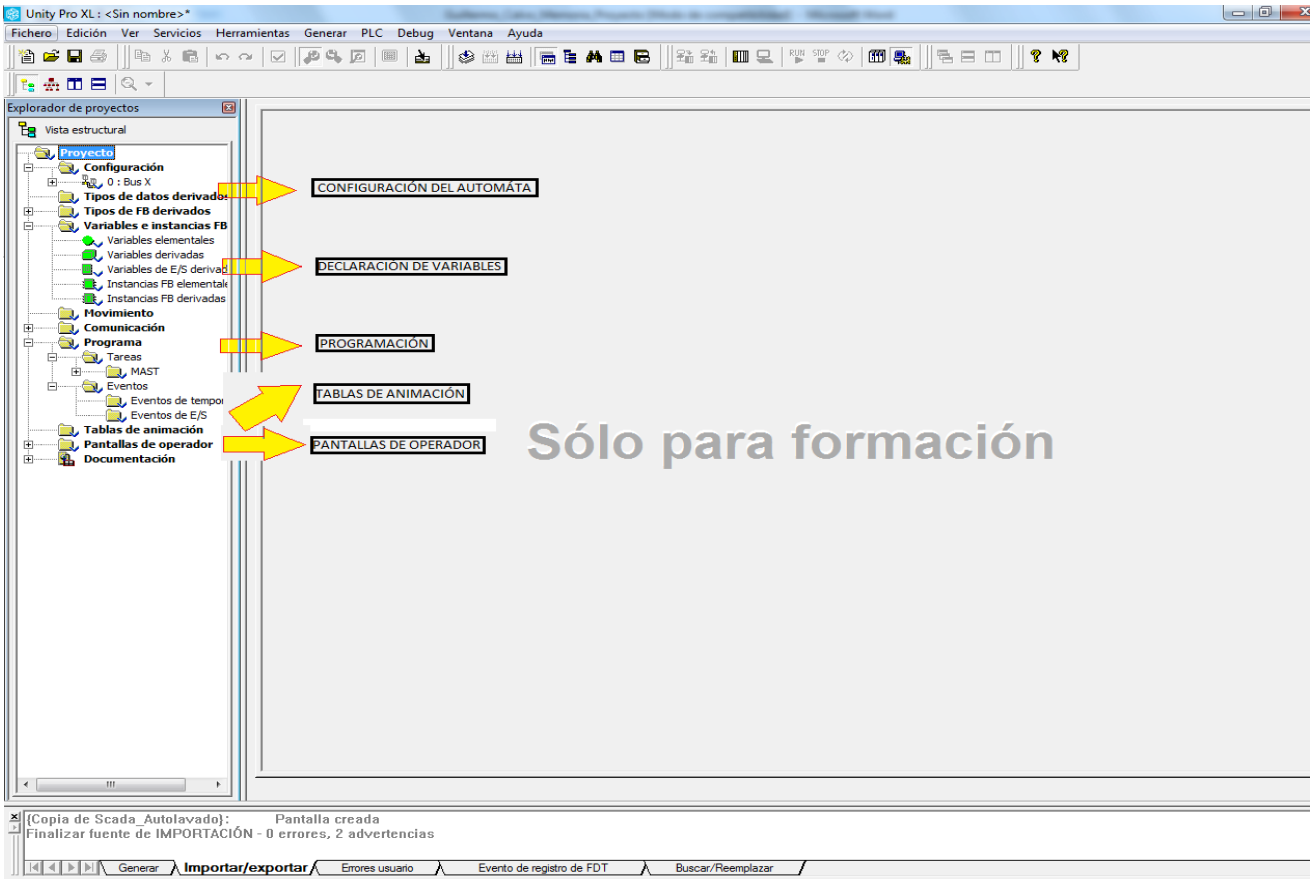


Figura 6: Explorador de Proyectos en Unity Pro

En el Explorador de Proyectos, mostrado en la Figura 6, el primer paso es definir el Hardware con el que se va a trabajar. Se necesita instalar los diferentes módulos en el Rack donde está colocada la CPU y una Fuente de Alimentación instalada automáticamente por el programa con un Rack.
Se puede observar en la Figura xx, que el Rack está dividido en varios Slots. La primera posición del Rack (la posición cero) estará ocupada por la CPU y a continuación se disponen las tarjetas necesarias. Este orden siempre deber ser seguido en la configuración del hardware, ya que si no se tiene en cuenta al compilar la aplicación, podría ocurrir un error.

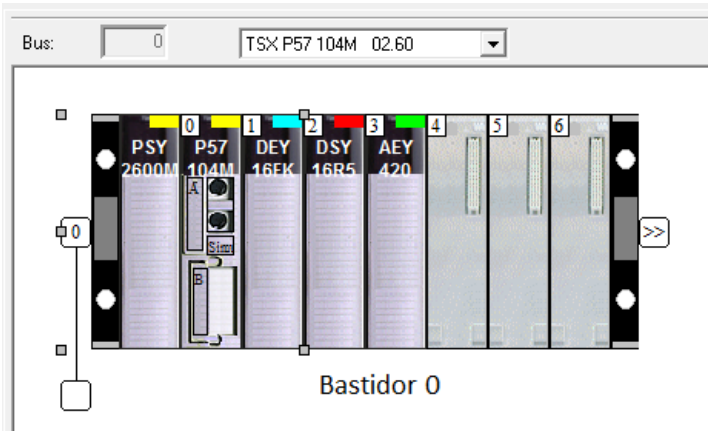


Figura 7: Configuración del Rack

El primero de los bloques del bastidor que contiene el ejemplo de la Figura 7, es el asignado con las siglas "PSY" corresponde a la Fuente de Alimentación de 100/240 VCA 26W. El bloque denominado "P57" es el procesador CPU. Las entradas digitales están contenidas en el bloque "DEY", y las salidas digitales en el bloque "DSY", y por último están el bloque de entradas analógicas "AEY".

A la hora de introducir direcciones del autómata estas se diferencian por el bastidor, slots del módulo y cenal, es decir, por los números que aparecen encima del bloque de las entradas y salidas (digitales y analógicas) esos representarán al Slot del módulo, y las direcciones dentro de cada módulo serán los canales. Un ejemplo:

- Para leer el valor de la entrada digital del canal 4, en el slot 1 del bastidor 0:
  - La dirección correcta sería: %I0.1.4
- Para leer el valor de la salida digital del canal 6, en el slot 2 del bastidor 0:
  - La dirección correcta sería: %Q0.2.6
- Para leer el valor de la entrada analógica del canal 1, en el slot 3 del bastidor 0:
  - La dirección correcta sería: %IW0.3.1

Se puede modificar esta configuración inicial en todos los aspectos, se pueden añadir nuevos bastidores, si se van a ocupar más entradas/salidas digitales/analógicas. Añadir un Rack mayor, un segundo Rack, módulos binarios, analógicos, conteo o de comunicación.

De entre todas las variables creadas, se puede hacer una clasificación especial según su uso: Hay algunas que son utilizadas para la programación del sistema, y por tanto también se aprovecharán para la simulación. Dichas variables corresponden con las entradas y salidas del sistema.

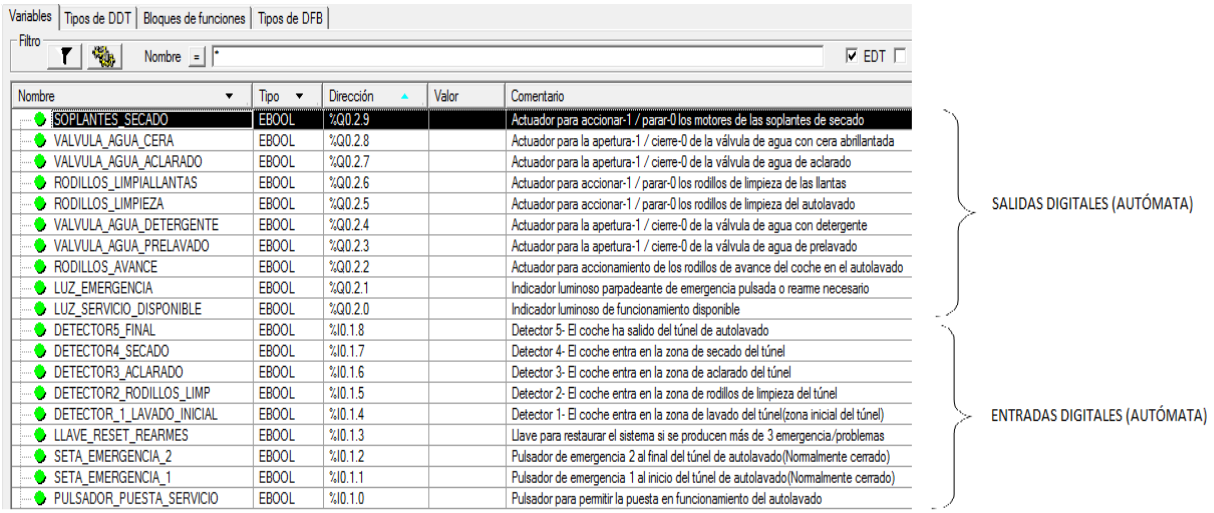
Sin embargo, hay algunas variables que exclusivamente han sido creadas para la simulación, y que más adelante se detallará su origen. Dichas variables son mostradas más adelante (Figura xx), y no se utilizan en código de la programación.

Las variables de entrada/salida digitales y analógicas del autómata son las que muestran la Figura 8 y Figura 9 respectivamente.

Una vez contempladas las variables, hay de definirlas. La manera de realizarlo, es escribiendo el nombre de cada variable, sin dejar espacios, e ir añadiendo los siguientes parámetros:

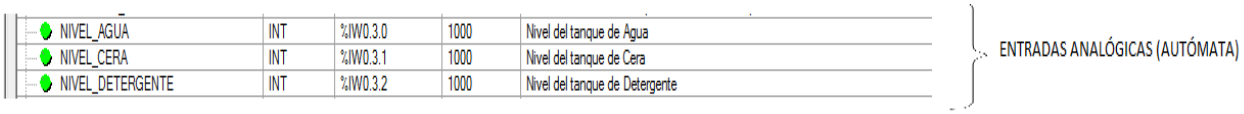
- Nombre
- Tipo
- Dirección
- Valor
- Comentario

Dichas variables se crean dentro del Explorador de Proyectos, en el apartado de "Variables e instancias FB" y una vez aquí, en variables elementales. El sistema permite hacer distinción entre "variables derivada", "variables derivadas de E/S", "Instancias FB elementales" e "Instancias FB derivadas".



Nombre	Tipo	Dirección	Valor	Comentario
SOPLANTES_SECADO	EBOOL	%Q0.2.9		Actuador para accionar-1 / parar-0 los motores de las soplantes de secado
VALVULA_AGUA_CERA	EBOOL	%Q0.2.8		Actuador para la apertura-1 / cierre-0 de la válvula de agua con cera abrillantada
VALVULA_AGUA_ACLARADO	EBOOL	%Q0.2.7		Actuador para la apertura-1 / cierre-0 de la válvula de agua de aclarado
RODILLOS_LIMPIALLANTAS	EBOOL	%Q0.2.6		Actuador para accionar-1 / parar-0 los rodillos de limpieza de las llantas
RODILLOS_LIMPIEZA	EBOOL	%Q0.2.5		Actuador para accionar-1 / parar-0 los rodillos de limpieza del autolavado
VALVULA_AGUA_DETERGENTE	EBOOL	%Q0.2.4		Actuador para la apertura-1 / cierre-0 de la válvula de agua con detergente
VALVULA_AGUA_PRELAVADO	EBOOL	%Q0.2.3		Actuador para la apertura-1 / cierre-0 de la válvula de agua de prelavado
RODILLOS_AVANCE	EBOOL	%Q0.2.2		Actuador para accionamiento de los rodillos de avance del coche en el autolavado
LUZ_EMERGENCIA	EBOOL	%Q0.2.1		Indicador luminoso parpadeante de emergencia pulsada o rearme necesario
LUZ_SERVICIO_DISPONIBLE	EBOOL	%Q0.2.0		Indicador luminoso de funcionamiento disponible
DETECTOR5_FINAL	EBOOL	%I0.1.8		Detector 5- El coche ha salido del túnel de autolavado
DETECTOR4_SECADO	EBOOL	%I0.1.7		Detector 4- El coche entra en la zona de secado del túnel
DETECTOR3_ACLARADO	EBOOL	%I0.1.6		Detector 3- El coche entra en la zona de aclarado del túnel
DETECTOR2_RODILLOS_LIMP	EBOOL	%I0.1.5		Detector 2- El coche entra en la zona de rodillos de limpieza del túnel
DETECTOR_1_LAVADO_INICIAL	EBOOL	%I0.1.4		Detector 1- El coche entra en la zona de lavado del túnel(zona inicial del túnel)
LLAVE_RESET_REARMES	EBOOL	%I0.1.3		Llave para restaurar el sistema si se producen más de 3 emergencia/problemas
SETA_EMERGENCIA_2	EBOOL	%I0.1.2		Pulsador de emergencia 2 al final del túnel de autolavado(Normalmente cerrado)
SETA_EMERGENCIA_1	EBOOL	%I0.1.1		Pulsador de emergencia 1 al inicio del túnel de autolavado(Normalmente cerrado)
PULSADOR_PUESTA_SERVICIO	EBOOL	%I0.1.0		Pulsador para permitir la puesta en funcionamiento del autolavado

Figura 8: Variables de entrada/salida digitales



NIVEL_AGUA	INT	%IW0.3.0	1000	Nivel del tanque de Agua
NIVEL_CERA	INT	%IW0.3.1	1000	Nivel del tanque de Cera
NIVEL_DETERGENTE	INT	%IW0.3.2	1000	Nivel del tanque de Detergente

Figura 9: Variables de entrada analógicas

Las variables que utilizamos para la simulación del proceso las mostramos en la siguiente figura. También utilizamos marcas internas para guardar estados de las etapas:

Variables | Tipos de DDT | Bloques de funciones | Tipos de DFB

Filtro — Nombre = * — EDT

Nombre	Tipo	Dirección	Valor	Comentario
PULSADOR_PUESTA_SERVICIO	EBOOL	%I0.1.0		Pulsador para permitir la puesta en funcionamiento del autolavado
VELOCIDAD_RAPIDA_SOPLANTE	INT			Velocidad rápida para las soplantes de secado
VELOCIDAD_LENTA_SOPLANTE	INT			Velocidad lenta para las soplantes de secado
TEMPORIZADOR_VELOCIDAD_RAP...	TIME			Temporización velocidad rápida para soplantes de secado (10 segundos)
TEMPORIZADOR_VELOCIDAD_LEN...	TIME			Temporización velocidad lenta para soplantes de secado (20 segundos)
TEMPORIZADOR_AGUA_PRELAVA...	TIME			Temporizador duración agua prelavado (10 segundos)
TEMPORIZADOR_AGUA_DETERGE...	TIME			Temporizador duración agua con detergente (20 segundos)
TEMPORIZADOR_AGUA_CERA	TIME			Temporización duración agua con cera (10 segundos)
TEMPORIZADOR_AGUA_ACLARADO	TIME			Temporizador duración agua aclarado (20 segundos)
SIM_DETECTOR5	EBOOL			Simulador D5 - el coche ha salido del túnel de autolavado
SIM_DETECTOR4	EBOOL			Simulador D4 - el coche entra en la zona de secado del túnel
SIM_DETECTOR3	EBOOL			Simulador D3 - el coche entra en la zona de aclarado del túnel
SIM_DETECTOR2	EBOOL			Simulador D2 - el coche entra en la zona de los rodillos de limpieza del túnel
SIM_DETECTOR1	EBOOL			Simulador D1 -el coche entra en la zona de lavado del túnel (zona inicial túnel)
SECADO_M_5_4	EBOOL			
SECADO_M_5_3	EBOOL			
SECADO_M_5_2	EBOOL			
SECADO_M_5_1	EBOOL			
SECADO_M_5_0	EBOOL			
SALIDA_TUNEL_M_6_1	EBOOL			
SALIDA_TUNEL_M_6_0	EBOOL			
RODILLOS_M_3_4	EBOOL			
RODILLOS_M_3_3	EBOOL			
RODILLOS_M_3_0	EBOOL			
Programa3_ON	STRING		Programa3	
PROGRAMA3	EBOOL			
PROGRAMA2	EBOOL			
Programa1_ON	STRING		Programa1	
PROGRAMA1	EBOOL			
Prgrama2_ON	STRING		Programa2	
PRELAVADO_M_2_2	EBOOL			
PRELAVADO_M_2_1	EBOOL			
PRELAVADO_M_2_0	EBOOL			
MW3	EBOOL			
MW2	EBOOL			
MW1	BOOL			
MW0	EBOOL			
MARCA_CICLO	WORD			Marca de ciclo
MANDO_CONTROL_M_1_2	EBOOL			
MANDO_CONTROL_M_1_1	EBOOL			
MANDO_CONTROL_M_1_0	EBOOL			
M_0_14	EBOOL			

Figura 10: Tabla de variables elementales

Para observar en el caso de las direcciones del autómata la nomenclatura de su dirección, dentro del "Explorador de Proyectos" en "Configuración del Bus" se hace doble clic en el bloque "DEY 16FK" del primer bastidor de la Figura 7, se obtiene la siguiente imagen de la Figura 11:

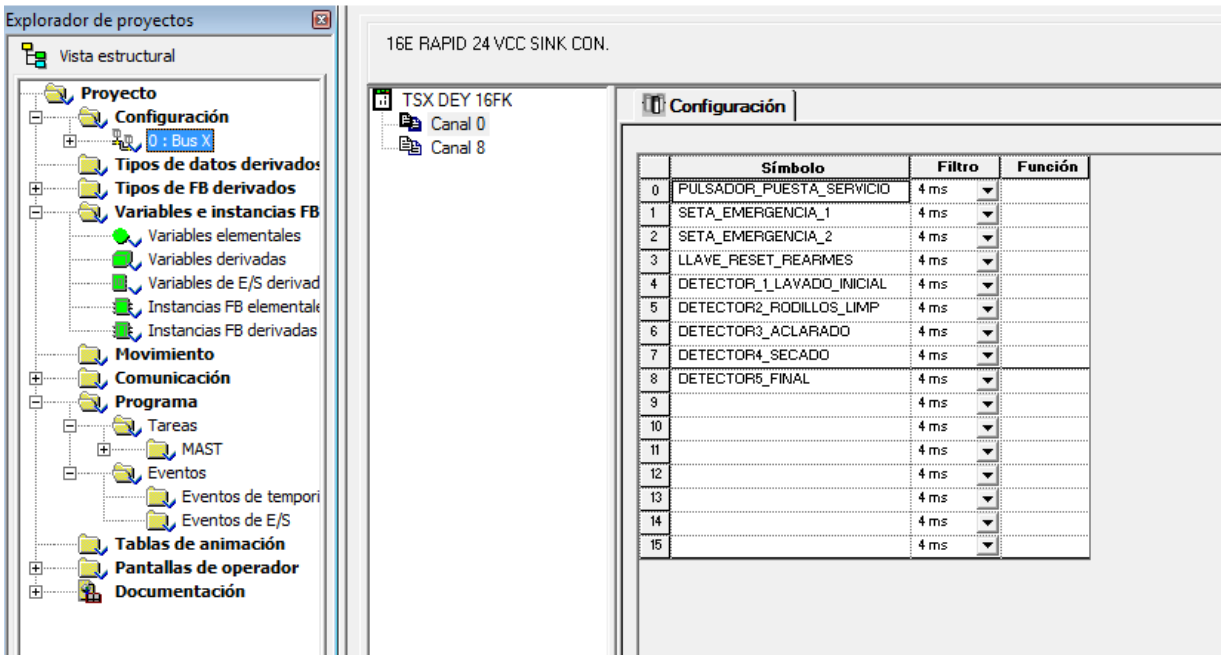


Figura 11: Entradas digitales bastidor

Haciendo doble clic en el bloque "DSY 16R5" del bastidor 0 de la Figura 7, se obtienen 16 salidas distribuidas en los Canales 0 y 8 de 1 byte cada uno. Dichos 16 bits aparecen en la Figura 12:

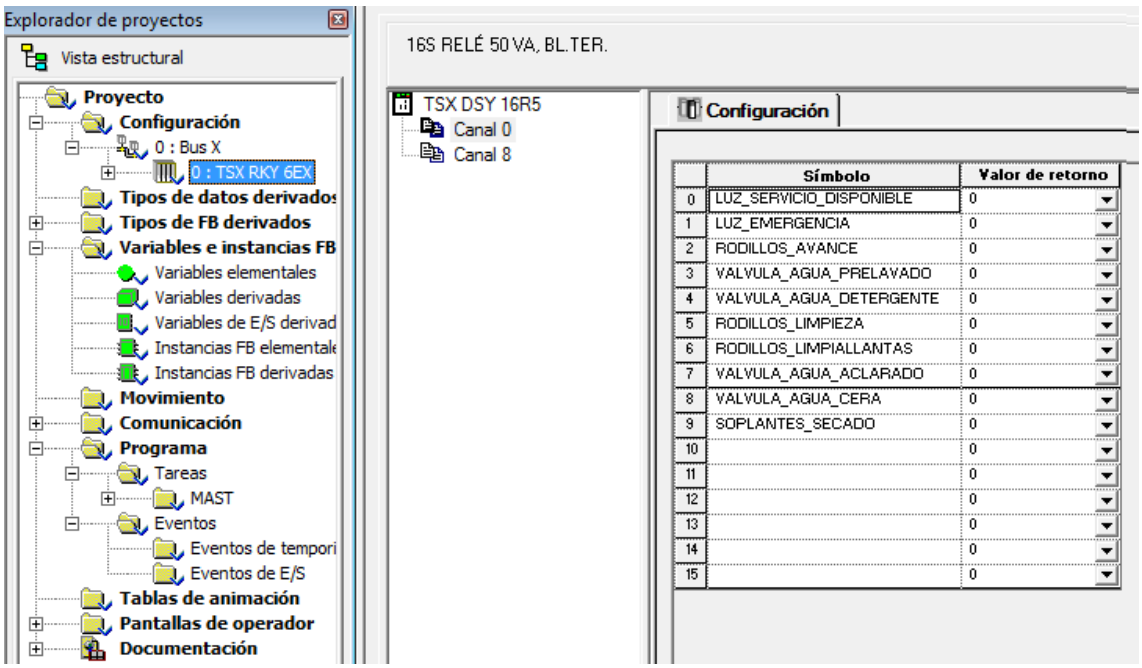


Figura 12: Salidas digitales bastidor

De la misma manera, hacemos doble click en el módulo de entradas analógicas "AEY 420", la cual se refleja en la siguiente figura 13:

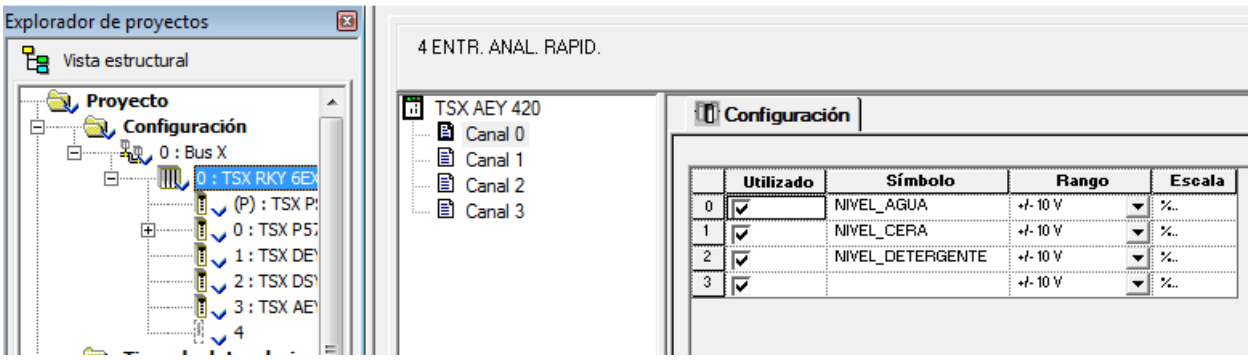


Figura 13: Entradas bastidor

4.3 Programación de la solución del problema

En este capítulo, se programa el sistema automatizado mediante un diagrama SFC. Es el método de descripción de procesos más visual y secuencial, mediante un gráfico funcional interpretable. Tal gráfico permite normalizar la forma de descripción del proceso. Por esta razón, no siempre es estrictamente necesario, pudiéndose prescindir de él y abordar directamente la programación. En este documento se desarrollará un diagrama SFC con sus respectivas etapas, y cada etapa programada en lenguajes diferentes que aporta el propio software: LD, IL, ST, para mostrar la versatilidad de Unity Pro.
Para definir un diagrama SFC, se comienza como muestra la Figura 14, creando una nueva sección SFC:

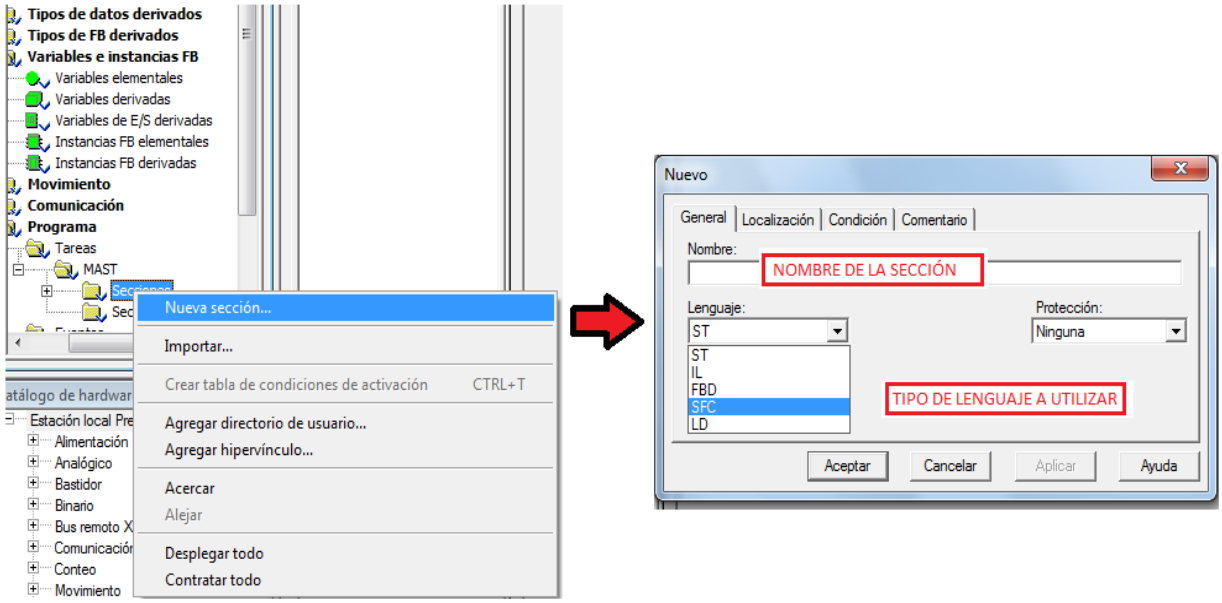


Figura 14: Creación de una sección SFC

El diagrama SFC final es el que se indica en la Figura 15. En él se observan definidas todas y cada una de las etapas que son necesarias para su ejecución.

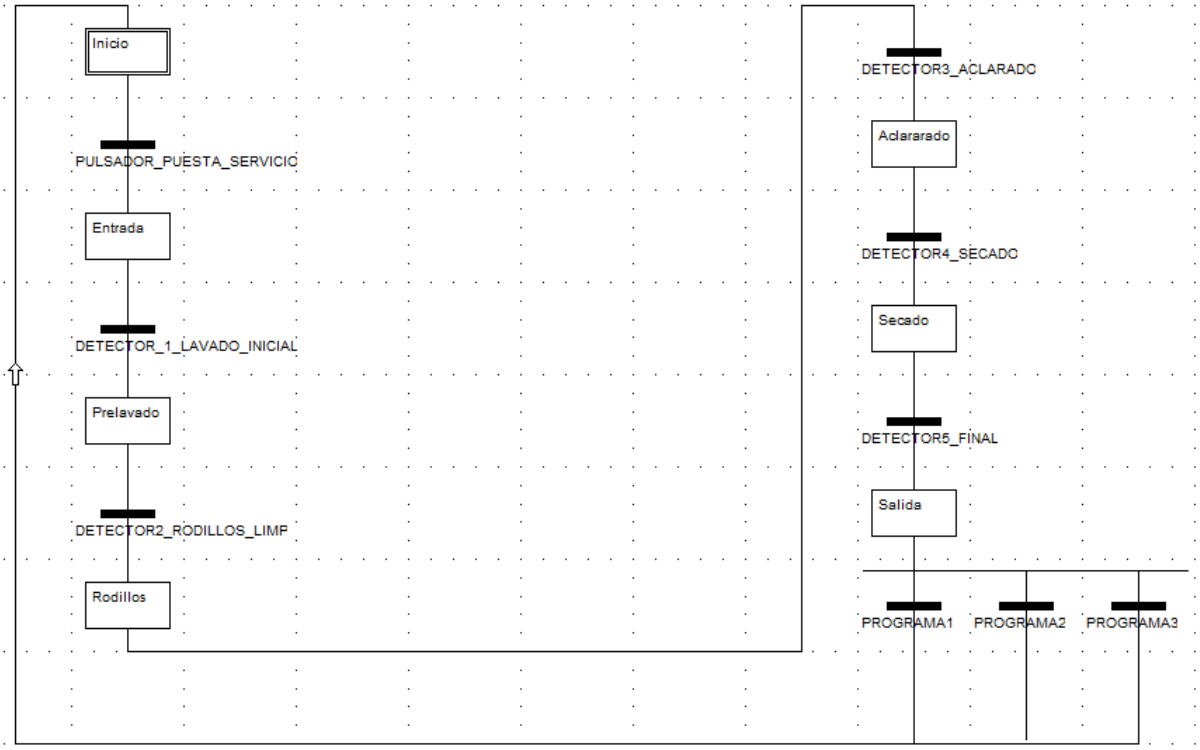


Figura 15: Diagrama SFC

Cuando se selecciona en el software Unity Pro una representación SFC, la pantalla presenta el siguiente aspecto de la Figura 16:

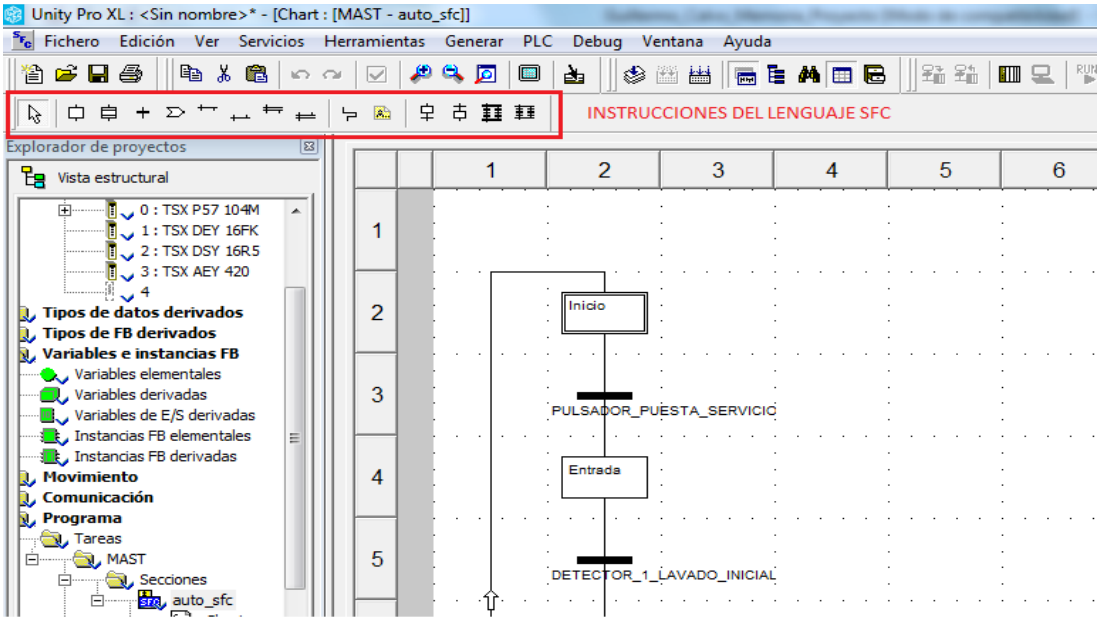


Figura 16: Instrucciones de lenguaje SFC

Y la barra de herramientas con sus respectivos significados quedan representados en la siguiente figura 17:

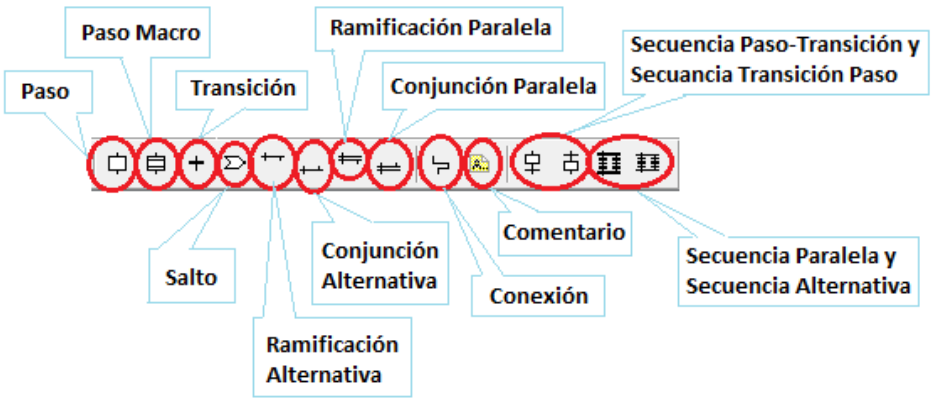


Figura 17: Elementos SFC

4.3.1 Programación SFC

Para la programación SFC de las etapas o pasos, todos los elementos se insertan arrastrando con el ratón uno a uno los símbolos necesitados. Una vez situados en la zona de trabajo (formada por una cuadrícula), se nombran y se programan las propiedades de cada paso o etapa, como muestra la Figura 18:

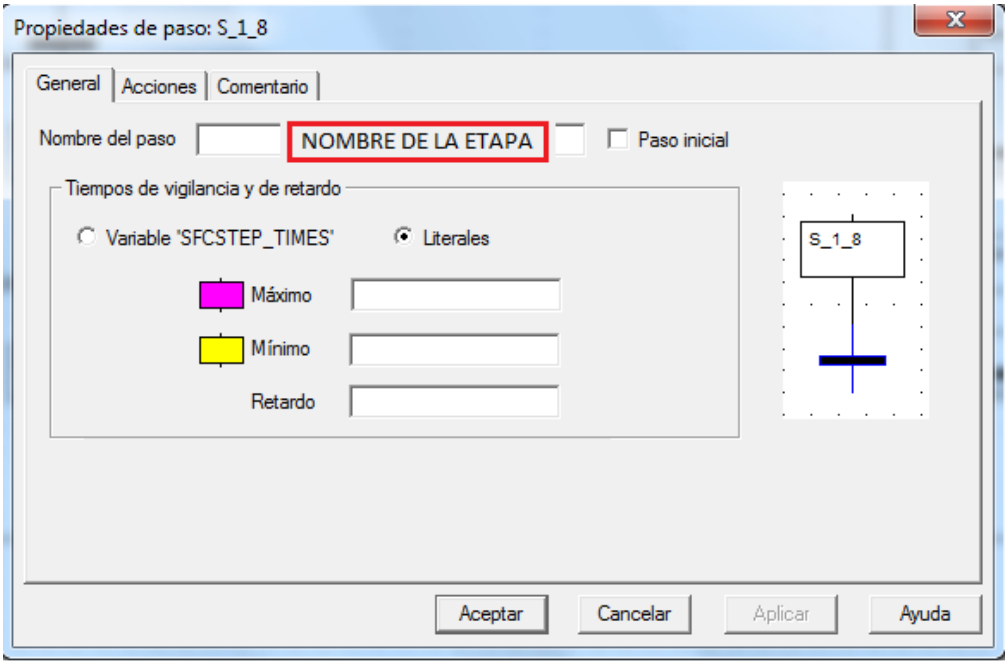


Figura 18: Propiedades de paso SFC

En la pestaña Acciones, es el lugar donde se programan las acciones que se deberá ejecutar cuando la etapa esté activa. Existen dos maneras de programar las acciones de los pasos:

- Acción ligada a una variable
- Acción ligada a una sección

Las acciones disponen de un descriptor que es el que determina, o bien en qué momento se ejecuta la acción, o qué acción se va a ejecutar. Algunos de los descriptores más utilizados:

- N: Se ejecuta la acción programada en la sección (o pone a "1" la variable asignada) durante el tiempo que está activa la etapa.
- P1: Se ejecuta la acción programada en la sección (o pone a "1" la variable asignada) cuando la etapa pasa de inactiva a activa (acción al activar).
- P0: Se ejecuta la acción programada en la sección (o pone a "1" la variable asignada) cuando la etapa pasa de activa a inactiva (acción al desactivar).
- S: Pone a "1" la variable asignada.
- R: Pone a "0" la variable asignada.

Para programar una acción ligada a una variable se selecciona la pestaña "Variable" en el recuadro acción de la pestaña "Acciones" y se selecciona dicha variable a través del desplegable que aparece al pulsar sobre el cuadrado gris. También hay que definir uno de los descriptores que se ajuste a la acción que se quiere ejecutar en esa etapa. Un ejemplo sería el que se representa en la figura 19:

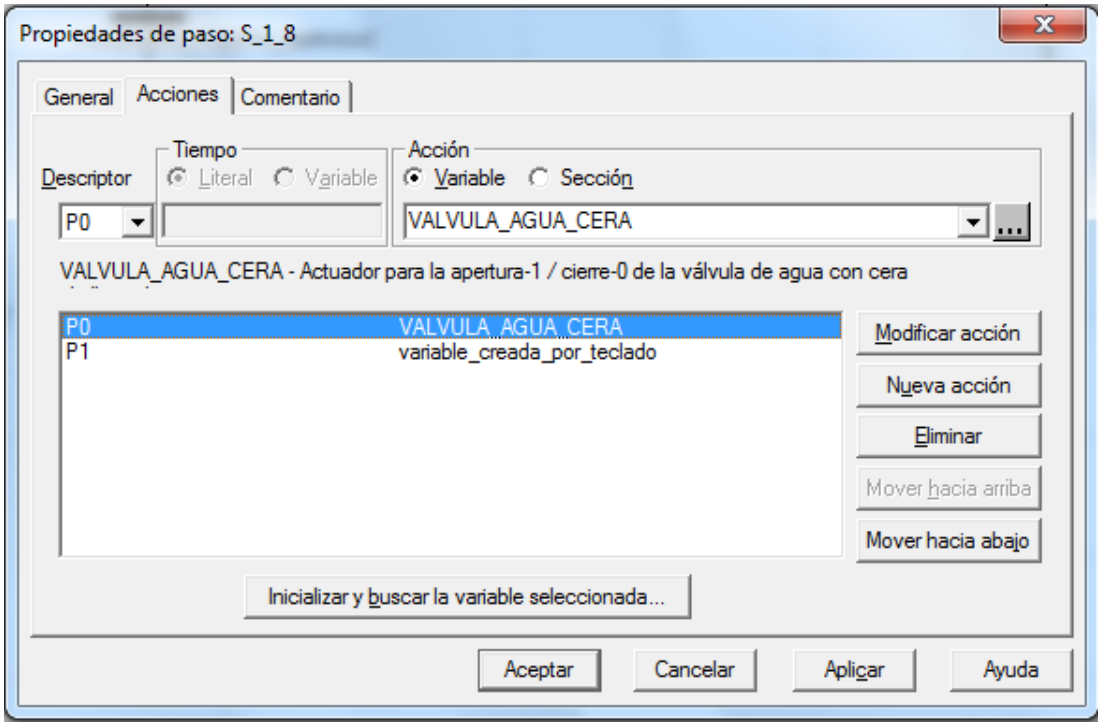


Figura 19: Acción ligada a una variable de paso

Se han creado dos variables asociadas a una acción, en la imagen se ve los dos tipos que puede haber, una que ya existe "VALVULA_AGUA_CERA" y otra que se ha creado introduciéndola por teclado.

Para determinar una acción ligada a una sección, se elige un descriptor que mejor se ajuste al programa, se pone nombre a la sección y se selecciona" Nueva acción". Tras pulsar el botón inferior de "Editar sección de acciones" mostrado en la Figura 20, el software mostrará una pantalla, en la que se podrá elegir uno de los lenguajes entre los cuatro disponibles, y en el que se quiere programar, tal y como se muestra en la Figura 21:

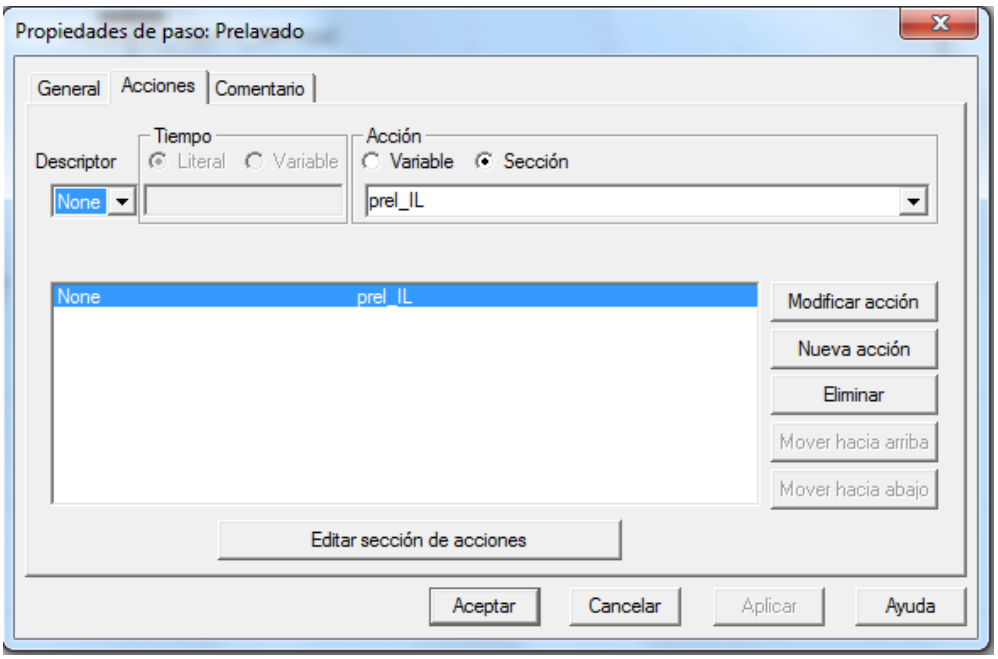


Figura 20: Acción ligada a una sección de paso

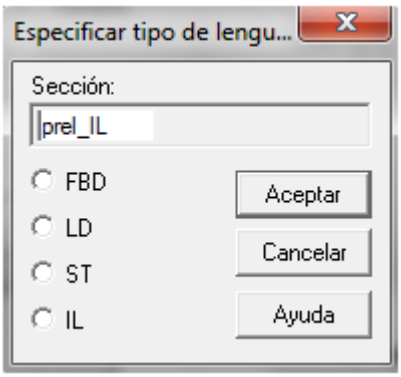


Figura 21: Tipo de lenguaje de la sección seleccionada

4.3.2 Programación de transiciones

Al igual que la programación de etapas, para programar las transiciones, existen dos formas posibles:

- La condición de transición ligada a una variable.
- La condición de transición ligada a una sección.

Para programar una condición de transición ligada a una variable, como muestra la Figura 22, es necesario seleccionar "Variable" en el tipo de condición de transición, y elegir la dirección, valor o variable EBool que hace referencia la transición:

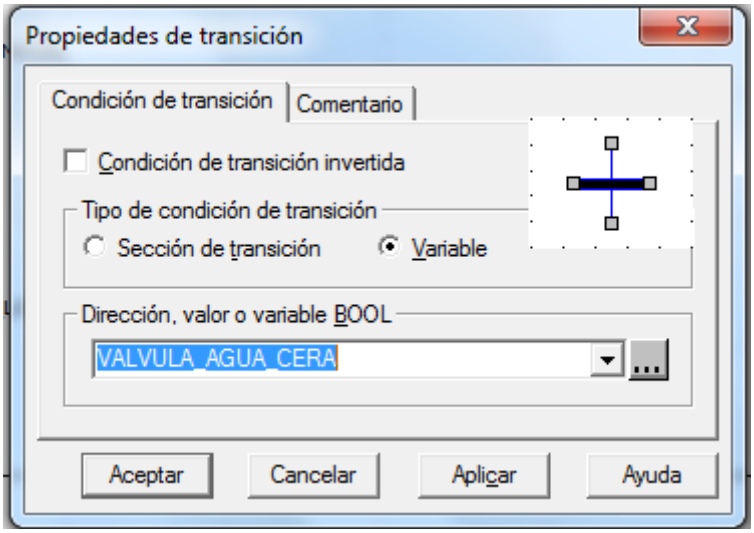


Figura 22: Propiedades de una variable de transición

Para programar la condición de transición ligada a una sección, es necesario seleccionar y poner el nombre en la "Sección de transición", como se visualiza en la Figura 23, y a continuación, pulsar "Editar". Aparece en la Figura 24, la pantalla donde se especifica el lenguaje de programación:

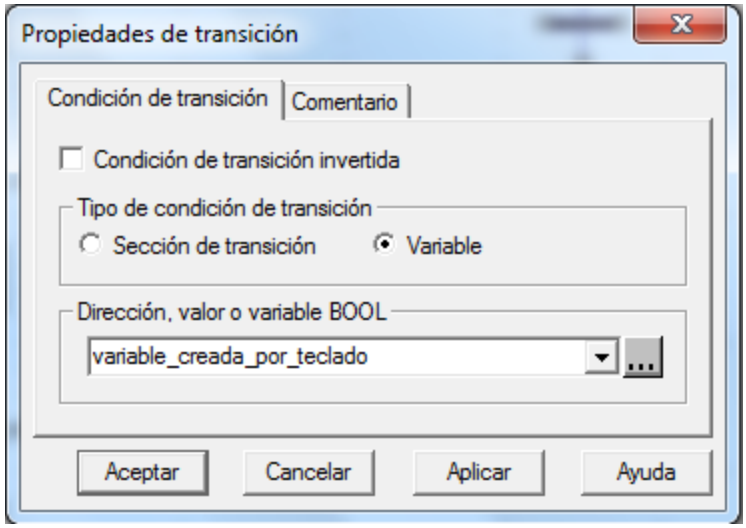


Figura 23: Variable de transición creada por teclado

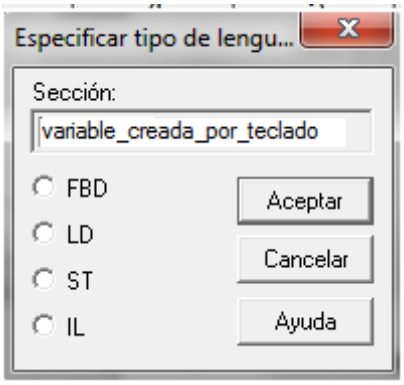


Figura 24: Lenguaje de programación de la variable de transición

4.3.3 Lenguaje ST

La principal característica del lenguaje estructurado (ST) es su gran potencia y versatilidad, siendo el más apropiado para funciones complejas y aplicaciones en las cuales sea necesario un programa optimizado. Es un lenguaje parecido al C, por tanto cualquier persona con cocimientos de programación informática es capaz de realizar aplicaciones complejas mediante el software Unity Pro. [10]

En esta sección se va a describir como se programa en lenguaje ST aquellas acciones y/o transiciones ligadas a secciones del diagrama SFC.
La Figura 25, recuerda los pasos a seguir para llegar a programar en lenguaje ST, para la condición de transición ligada a una sección. Tras este paso, aparecerá una pantalla en la que es posible escribir en ST.

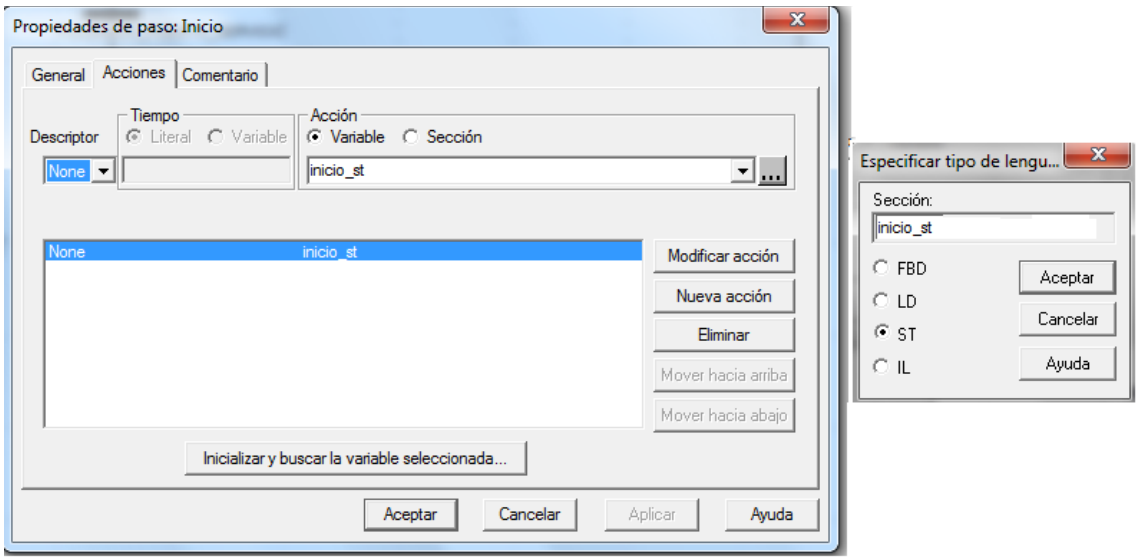


Figura 25: Variable creada para la etapa Inicio en lenguaje ST

ST es un lenguaje escrito, una vez creada la acción toca programar en el lenguaje que hemos elegido, para este caso tenemos esta serie de instrucciones.

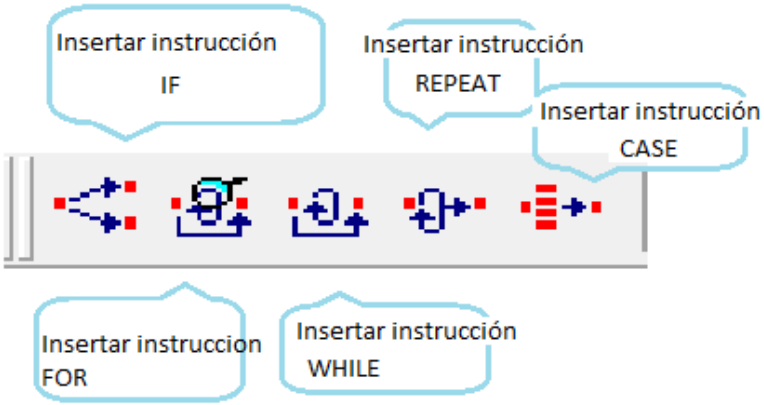


Figura 26: Instrucciones lenguajes ST

La parte más importante de la barra de herramientas son las instrucciones de control. Gracias a ellas podemos ir creando la aplicación paso a paso. A continuación quedan definidas:

- *IF:* La instrucción IF determina que una instrucción o un grupo de instrucciones se ejecuten sólo si la expresión booleana correspondiente tiene el valor 1. Si la condición es 0, la instrucción o el grupo de instrucciones no se ejecuta. "IF…THEN…END_IF,ELSE y ELSIF…THEN"

- *FOR*: La instrucción FOR repite una secuencia de instrucciones hasta la instrucción END_FOR. La cantidad de repeticiones se determina mediante el valor inicial, el valor final y la variable de control. "FOR…TO…BY…DO…END_FOR"

- *WHILE:* La instrucción WHILE provoca la ejecución repetitiva de una secuencia de instrucciones hasta que la expresión booleana correspondiente sea 0. Si la expresión es falsa desde el principio, no se ejecuta el grupo de instrucciones." WHILE…DO…END WHILE".

- *REPEAT*: La instrucción REPEAT provoca la ejecución repetitiva de una secuencia de instrucciones (al menos una vez) hasta que la condición booleana correspondiente sea 1. "REPEAT…UNTIL…END_REPEAT"

- *CASE*: La instrucción CASE está compuesta por una expresión del tipo datos INT (el "selector") y una lista de grupos de instrucciones. Cada grupo provisto de una etiqueta que está compuesta por uno o más números enteros (INT,DINT, UINT,UDINT) o rangos de valores enteros. Se ejecuta el primer grupo de instrucciones cuya etiqueta contenga el valor calculado del selector. De lo contario, no se ejecuta ninguna de las instrucciones. "CASE…OF…END_CASE".

Para nuestro proceso automatizado, en la etapa donde he utilizado el lenguaje ST ha sido en el **mando de control**, a continuación se muestra el código que se ha utilizado:

```
(*MANDO DE CONTROL*)

MANDO_CONTROL_M_1_2:=NOT LUZ_SERVICIO_DISPONIBLE;

        IF PULSADOR_PUESTA_SERVICIO AND NOT SETA_EMERGENCIA_1 AND NOT SETA_EMERGENCIA_2
        OR MW0 OR MW2 OR MW3
        THEN LUZ_SERVICIO_DISPONIBLE:=1;
        END_IF;

        IF PULSADOR_PUESTA_SERVICIO AND NOT SETA_EMERGENCIA_1 AND NOT SETA_EMERGENCIA_2
        THEN DETECTOR1_OP:=1;
        END_IF;

        IF PULSADOR_PUESTA_SERVICIO AND NOT SETA_EMERGENCIA_1 AND NOT SETA_EMERGENCIA_2
        THEN MANDO_CONTROL_M_1_0:=0;
        END_IF;



        IF SETA_EMERGENCIA_1 OR SETA_EMERGENCIA_2
        THEN MANDO_CONTROL_M_1_0:=1;
        END_IF;

        LUZ_EMERGENCIA:=MANDO_CONTROL_M_1_0 AND %S4;

        IF MANDO_CONTROL_M_1_0
        THEN LUZ_SERVICIO_DISPONIBLE:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN VALVULA_AGUA_DETERGENTE:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN VALVULA_AGUA_PRELAVADO:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN RODILLOS_AVANCE:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN RODILLOS_LIMPIALLANTAS:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN RODILLOS_LIMPIEZA:=0;
        END_IF;
```

1/2

```
        IF MANDO_CONTROL_M_1_0
        THEN MANDO_CONTROL_M_1_1:=0;
        END_IF;

        IF MANDO_CONTROL_M_1_0
        THEN SOPLANTES_SECADO:=0;
        END_IF;

        M_0_10:=SETA_EMERGENCIA_1 OR SETA_EMERGENCIA_2;


FBI_15 (EN:=MANDO_CONTROL_M_1_0 (*BOOL*),
        CU :=M_0_10 (*BOOL*),
        R := LLAVE_RESET_REARMES(*BOOL*),
        PV := 3(*INT*),
        CV => CONTADOR_REARMES (*INT*));


        IF PROGRAMA1
        THEN MW0:=1;
        END_IF;

        IF PROGRAMA2
        THEN MW3:=1;
        END_IF;

        IF PROGRAMA3
        THEN MW2:=1;
        END_IF;

        M_0_11:=PULSADOR_PUESTA_SERVICIO AND MW0;

(*ANY*)Lector_Programa := MOVE (EN:= M_0_11,
                               IN :=Programa1_ON (*ANY*));


        M_0_12:=PULSADOR_PUESTA_SERVICIO AND MW3;

(*ANY*)Lector_Programa := MOVE (EN:= M_0_12,
                               IN :=Prgrama2_ON (*ANY*));

        M_0_14:=PULSADOR_PUESTA_SERVICIO AND MW2;

(*ANY*)Lector_Programa := MOVE (EN:= M_0_14,
                               IN :=Programa3_ON (*ANY*));
```

2/2

Figura 27: Código del Mando de control en lenguaje ST

El mando de control de lo que se encarga básicamente es de poner en marcha nuestro proceso automatizado, tiene control sobre las setas de emergencia, los programas establecidos y la puesta en servicio de la estación. A través del bloque de función “MOVE” transferimos a nuestro cuadro de mandos la visualización del programa elegido por el usuario, para que en todo momento se dé constancia de lo que se ha elegido.

4.3.4 Lenguaje IL

El lenguaje IL es un tipo de lenguaje ensamblador con un repertorio muy reducido de instrucciones. [11]

Los programas utilizan un estilo muy similar al empleado por los lenguajes de ensamblador. Este tipo de lenguaje es una trascripción elemental e inmediata de las instrucciones del lenguaje máquina, las cuales están representadas por expresiones nemotécnicas.

Este tipo de lenguaje se suele emplear para pequeñas aplicaciones y sobre todo para optimizar partes de una aplicación.

La semántica y operadores que se utilizan en este lenguaje vienen reflejados en la siguiente tabla:

LD	DEFINE EL RESULTADO ACTUAL IGUAL A OPERANDO	ADD	SUMA
ST	RESULTADO ACTUAL	SUB	RESTA
S	SISTEMA BOOLEANO:PONE A 1 LA INSTRUCCIÓN	MUL	MULTIPLICACIÓN
R	SISTEMA BOOLEANO:PONE A 0 LA INSTRUCCIÓN	DIV	DIVISIÓN
JMP	SALTO DE ETIQUETA	GT	COMPARACIÓN:>
CAL	LLAMADA AL BLOQUE DE FUNCIÓN	GE	COMPARACIÓN:>=
RET	DUVUELVE LA LLAMADA A LA FUNCIÓN	EQ	COMPARACIÓN:=
&,AND	BOOLEANO AND	NE	COMPARACIÓN:<>
OR	BOOLEANO OR	LE	COMPARACIÓN:<=
XOR	BOOLEANO XOR	LT	COMPARACIÓN:<

Tabla 1: Instrucciones lenguaje IL

Un ejemplo para a continuación poder comprender las etapas programadas en este lenguaje viene explicado en la siguiente Figura 28:

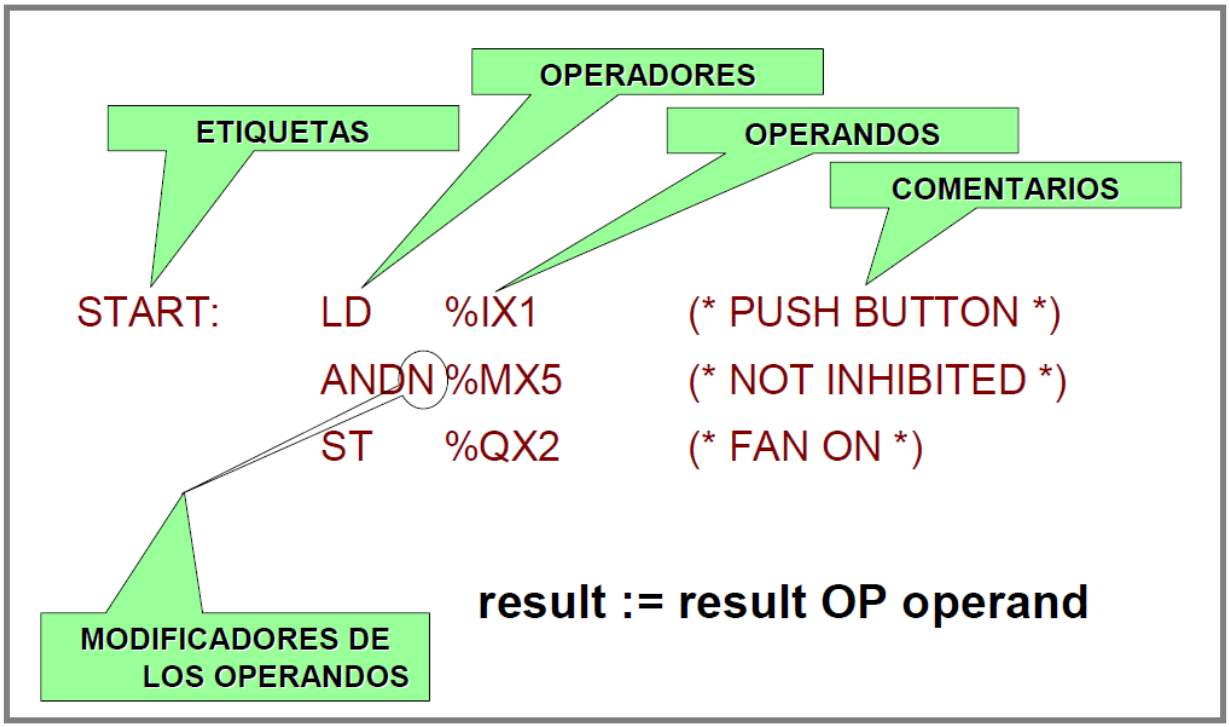


Figura 28: Ejemplo estructura lenguaje IL

Para nuestro proceso de automatización, los procesos que han sido programados en este lenguaje han sido la etapa de prelavado y detergente y la etapa de rodillos, se han utilizado temporizadores y contadores en ambas etapas. De esta forma el resultado es el siguiente:

```
(*PRELAVADO*)

LD LUZ_SERVICIO_DISPONIBLE

AND DETECTOR_1_LAVADO_INICIAL
ANDN VALVULA_AGUA_ACLARADO
ANDN VALVULA_AGUA_CERA
ANDN RODILLOS_LIMPIEZA
ANDN SOPLANTES_SECADO


S RODILLOS_AVANCE
S PRELAVADO_M_2_0
R DETECTOR1_OF
R MW0
R MW3
R MW2

CAL FBI_13 (EN:= LUZ_SERVICIO_DISPONIBLE (*EBOOL*),
           IN :=PRELAVADO_M_2_0 (*EBOOL*),
           PT :=t#8s (*TIME*),
           Q =>M_0_0 (*EBOOL*))


LD M_0_0
AND LUZ_SERVICIO_DISPONIBLE
ST VALVULA_AGUA_PRELAVADO

LD M_0_0
ANE LUZ_SERVICIO_DISPONIBLE
S PRELAVADO_M_2_1

LD M_0_0
AND LUZ_SERVICIO_DISPONIBLE
R PRELAVADO_M_2_0

LD PRELAVADO_M_2_1
ANDN VALVULA_AGUA_PRELAVADO
ST M_0_2

CAL FBI_14 (EN:= LUZ_SERVICIO_DISPONIBLE (*EBOOL*),
           IN := M_0_2 (*EBOOL*),
           PT :=t#12s (*TIME*),        (*TEMPORIZADOR
           Q =>M_0_1 (*EBOOL*))
```

1/2

```
LD M_0_1
AND LUZ_SERVICIO_DISPONIBLE
ST VALVULA_AGUA_DETERGENTE

LD M_0_1
AND LUZ_SERVICIO_DISPONIBLE
S PRELAVADO_M_2_2

LD M_0_1
AND LUZ_SERVICIO_DISPONIBLE
R PRELAVADO_M_2_1

LD VALVULA_AGUA_PRELAVADO
OR VALVULA_AGUA_ACLARADO
ST M_0_6

SUB_REAL    (EN:= M_0_6,
             IN1 := NIVEL_AGUA(*REAL*),
             IN2 := 0.2(*REAL*))
             ST NIVEL_AGUA(*REAL*)

LD VALVULA_AGUA_PRELAVADO
OR VALVULA_AGUA_DETERGENTE
ST M_0_7

SUB_REAL    (EN:= M_0_7,
             IN1 := CINTA_TRANSPORT(*REAL*),
             IN2 := 0.23(*REAL*))              (*OPERACIÓN MATEMÁTICA*)
             ST CINTA_TRANSPORT(*REAL*)

SUB_REAL    (EN:= VALVULA_AGUA_DETERGENTE,
             IN1 := NIVEL_DETERGENTE(*REAL*),  (*OPERACIÓN MATEMÁTICA*)
             IN2 := 0.1(*REAL*))
             ST NIVEL_DETERGENTE(*REAL*)

LD VALVULA_AGUA_PRELAVADO
OR VALVULA_AGUA_DETERGENTE
AND RODILLOS_AVANCE
AND %S6
ST M_0_8

ADD_REAL (EN:= M_0_8,
          IN1 :=0.01 (*REAL*),
          IN2 :=DISTANCIA_RECORRIDA (*REAL*)) (*OPERACIÓN MATEMÁTICA*)
          ST DISTANCIA_RECORRIDA(*REAL*)
```

2/2

```
LD PULSADOR_PUESTA_SERVICIO
ANDN SETA_EMERGENCIA_1
ANDN SETA_EMERGENCIA_2
S LUZ_SERVICIO_DISPONIBLE
S DETECTOR1_OF
R MANDO_CONTROL_M_1_0

LD SETA_EMERGENCIA_1
OR SETA_EMERGENCIA_2
S MANDO_CONTROL_M_1_0


LD MANDO_CONTROL_M_1_0
R LUZ_SERVICIO_DISPONIBLE
R VALVULA_AGUA_DETERGENTE
R VALVULA_AGUA_PRELAVADO
R RODILLOS_AVANCE
R RODILLOS_LIMPIALLANTAS
R RODILLOS_LIMPIEZA
R MANDO_CONTROL_M_1_1
R SOPLANTES_SECADO

LD MANDO_CONTROL_M_1_0
AND %S4
ST LUZ_EMERGENCIA

LD SETA_EMERGENCIA_1
OR SETA_EMERGENCIA_2
ST M_0_10

CAL FBI_17 (EN:=MANDO_CONTROL_M_1_0 (*BOOL*),
        CU :=M_0_10 (*BOOL*),
        R := LLAVE_RESET_REARMES(*BOOL*),
        PV := 3(*INT*),
        CV => CONTADOR_REARMES (*INT*))

                (*CONTADOR*)
```

3/2

Figura 29: Código de la etapa Prelavado en lenguaje IL

```
(*RODILLOS*)

LD DETECTOR2_RODILLOS_LIMP

AND PRELAVADO_M_2_2
ANDN VALVULA_AGUA_DETERGENTE
ANDN VALVULA_AGUA_PRELAVADO
ANDN SOPLANTES_SECADO
ANDN VALVULA_AGUA_ACLARADO
ANDN VALVULA_AGUA_CERA

S RODILLOS_M_3_0
R PRELAVADO_M_2_2
S RODILLOS_M_3_3

LD LUZ_SERVICIO_DISPONIBLE
AND RODILLOS_M_3_3
ST M_0_3

CAL FBI_12 (EN:= M_0_3 (*EBOOL*),
           IN :=RODILLOS_M_3_0 (*EBOOL*),
           PT :=t#4s (*TIME*),
           Q =>M_0_4 (*EBOOL*))

LD M_0_4
AND LUZ_SERVICIO_DISPONIBLE
ST RODILLOS_LIMPIALLANTAS

LD M_0_4
AND LUZ_SERVICIO_DISPONIBLE
ST RODILLOS_LIMPIEZA

LD M_0_4
AND LUZ_SERVICIO_DISPONIBLE
R RODILLOS_M_3_0

LD M_0_4
ANDN VALVULA_AGUA_PRELAVADO
S RODILLOS_M_3_4

SUB_REAL    (EN:= RODILLOS_LIMPIEZA,
             IN1 := CINTA_TRANSPORT(*REAL*),
             IN2 := 1.0(*REAL*))
             ST CINTA_TRANSPORT(*REAL*)

SUB_REAL    (EN:= RODILLOS_LIMPIEZA,
             IN1 := CINTA_TRANSPORT(*REAL*),
             IN2 := 0.25(*REAL*))
             ST CINTA_TRANSPORT(*REAL*)
```

1/2

```
LD RODILLOS_LIMPIEZA
AND RODILLOS_AVANCE
AND %S6
ST M_0_9

ADD_REAL (EN:= M_0_9,                          (*OPERACIÓN MATEMÁTICA*)
          IN1 :=0.01 (*REAL*),
          IN2 :=DISTANCIA_RECORRIDA (*REAL*))
          ST DISTANCIA_RECORRIDA(*REAL*)



LD PULSADOR_PUESTA_SERVICIO
ANDN SETA_EMERGENCIA_1
ANDN SETA_EMERGENCIA_2
S LUZ_SERVICIO_DISPONIBLE
S DETECTOR1_OF
R MANDO_CONTROL_M_1_0

LD SETA_EMERGENCIA_1
OR SETA_EMERGENCIA_2
S MANDO_CONTROL_M_1_0


LD MANDO_CONTROL_M_1_0
R LUZ_SERVICIO_DISPONIBLE
R VALVULA_AGUA_DETERGENTE
R VALVULA_AGUA_PRELAVADO
R RODILLOS_AVANCE
R RODILLOS_LIMPIALLANTAS
R RODILLOS_LIMPIEZA
R MANDO_CONTROL_M_1_1
R SOPLANTES_SECADO

LD MANDO_CONTROL_M_1_0
AND %S4
ST LUZ_EMERGENCIA

LD SETA_EMERGENCIA_1
OR SETA_EMERGENCIA_2
ST M_0_10

CAL FBI_16 (EN:=MANDO_CONTROL_M_1_0 (*BOOL*),
        CU :=M_0_10 (*BOOL*),
        R := LLAVE_RESET_REARMES(*BOOL*),
        PV := 3(*INT*),
        CV => CONTADOR_REARMES (*INT*))
           ST DISTANCIA_RECORRIDA(*REAL*)
```

2/2

Figura 30: Código de la etapa Rodillos en lenguaje IL

4.3.5 Lenguaje LD

La lógica de la escalera o ladder es el lenguaje de programación más utilizado para la programación de los PLC´s. Este lenguaje es con el que se empezó a programar, de ahí que haya cierta semejanza con los diagramas eléctricos.
Este lenguaje está especialmente indicado para facilitar el cambio de un sistema de control realizado con relés por un PLC. [12]

Para programar una etapa de nuestro proceso automatizado en este tipo de lenguaje seguiremos esta secuencia:

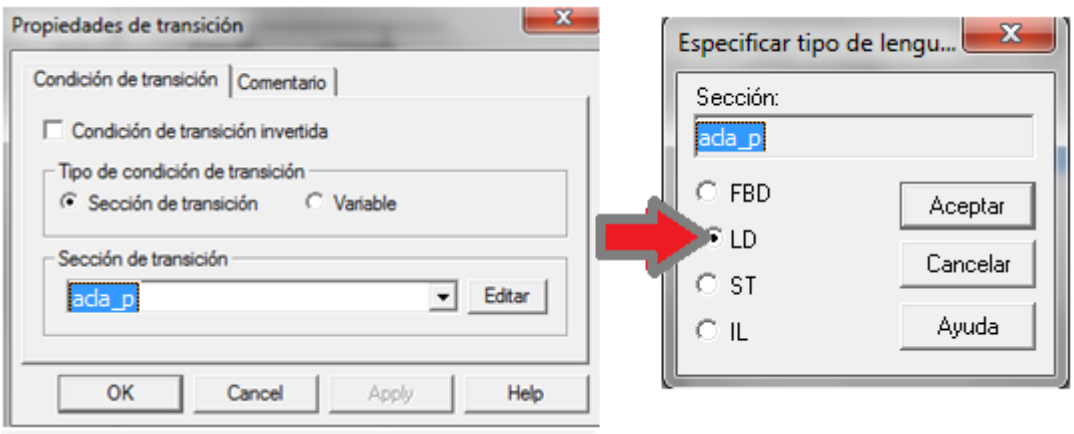


Figura 31: Creación de etapa con lenguaje LD

A continuación definimos nuestra acción ligada a la sección de transición:

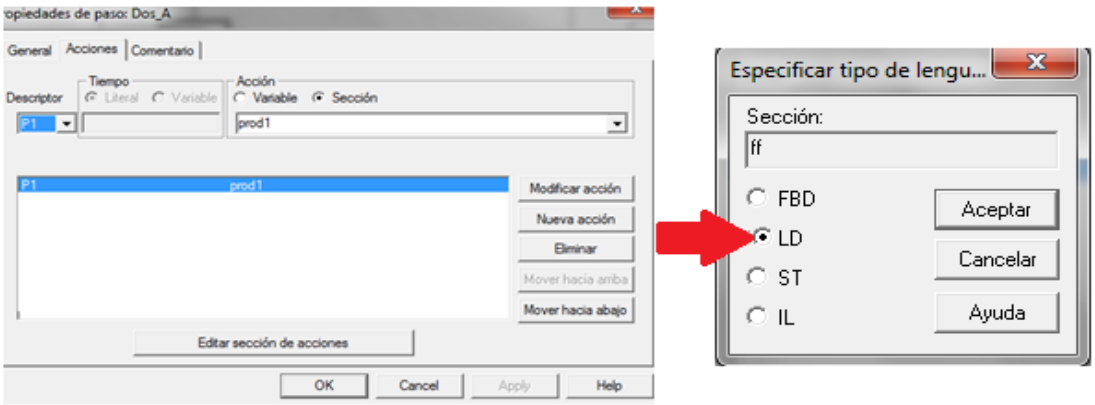


Figura 32: Definición de la acción ligada a la sección

Las herramientas que podemos utilizar para este lenguaje vienen establecidas en nuestra barra de herramientas:

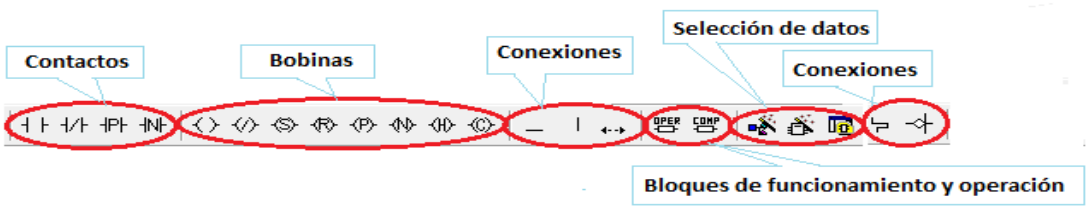


Figura 33: Instrucciones lenguaje LD

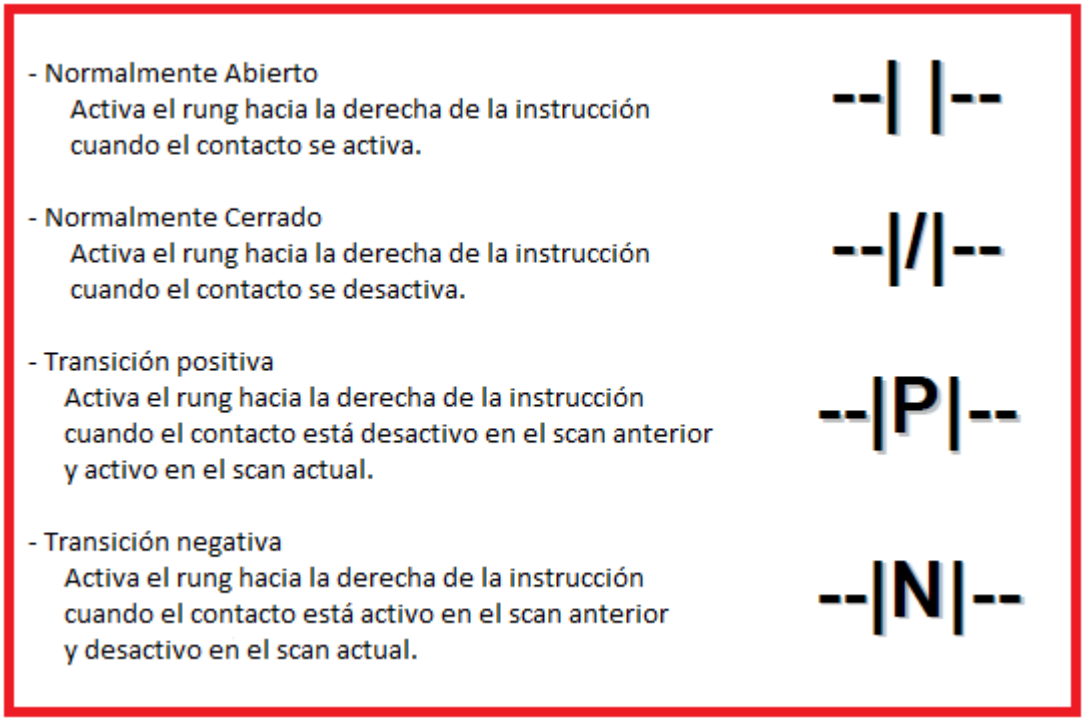


Figura 34: Definición de los campos de entrada del lenguaje LD

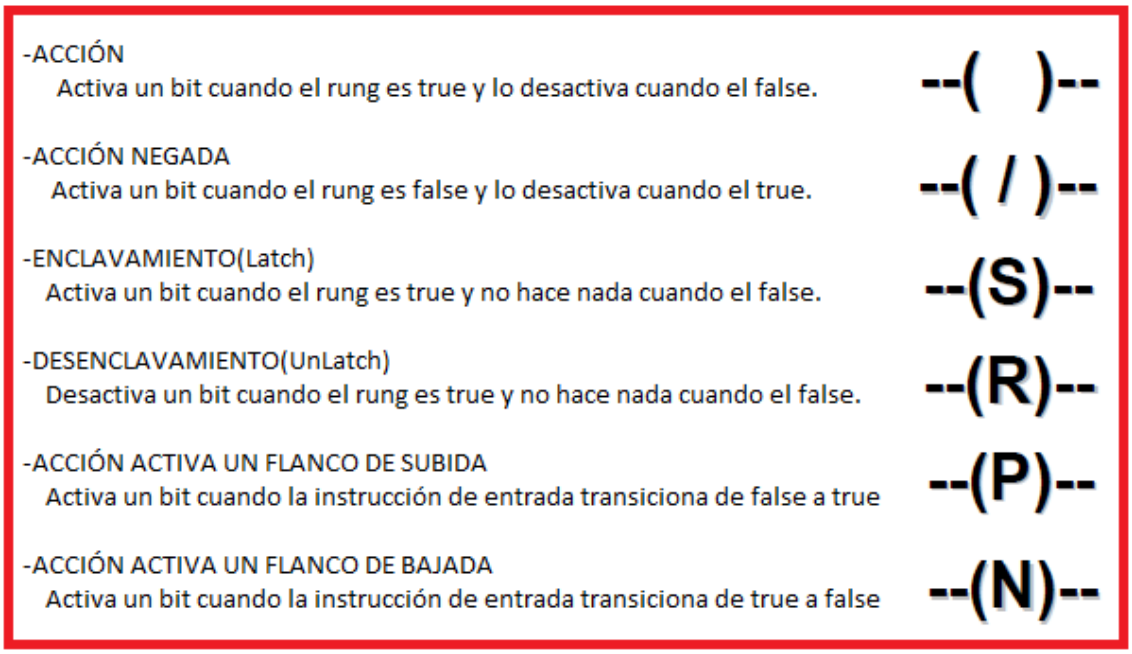


Figura 35: Definición de los campos de salida del lenguaje LD

Para las etapas de Aclarado, Secado y control de niveles de nuestra simulación hemos optado por programarlo con el lenguaje LD, de tal manera que nos ha quedado así:

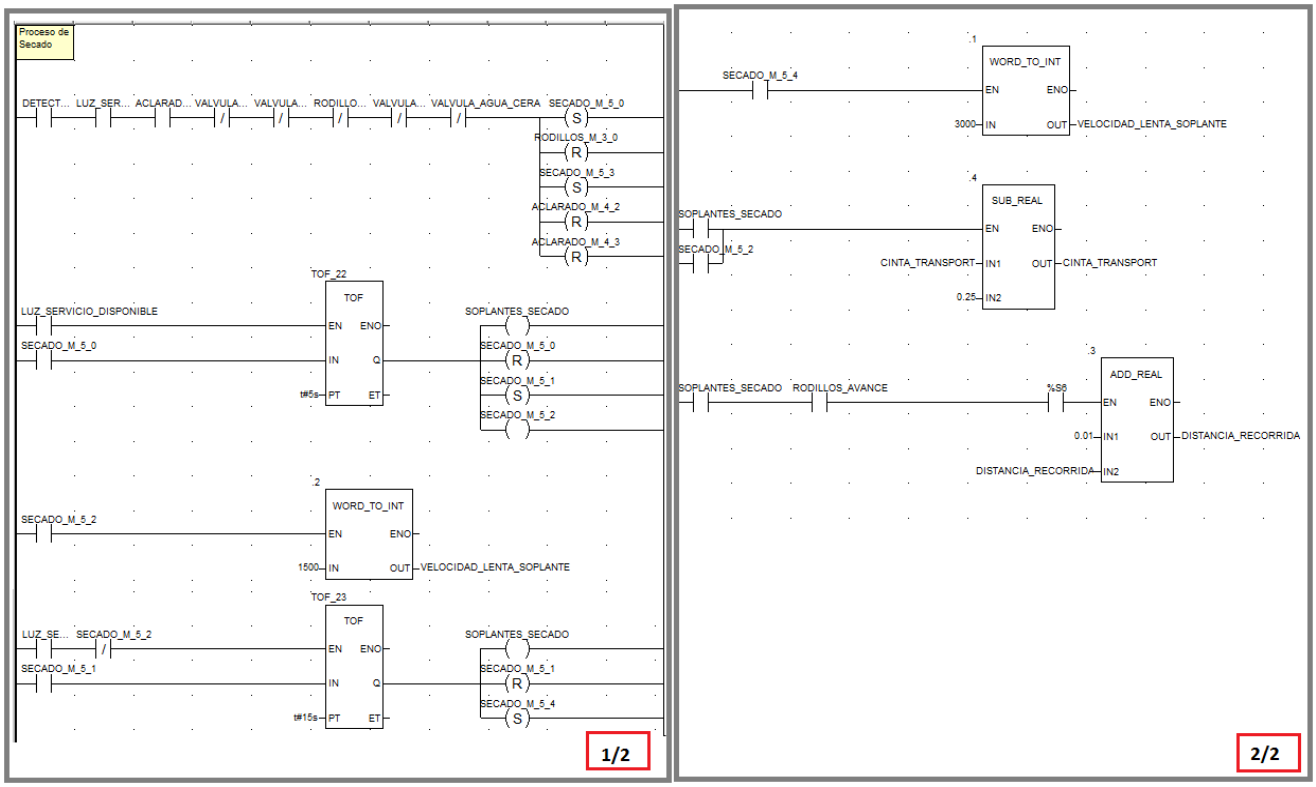


Figura 36: Código de la etapa Secado en lenguaje LD

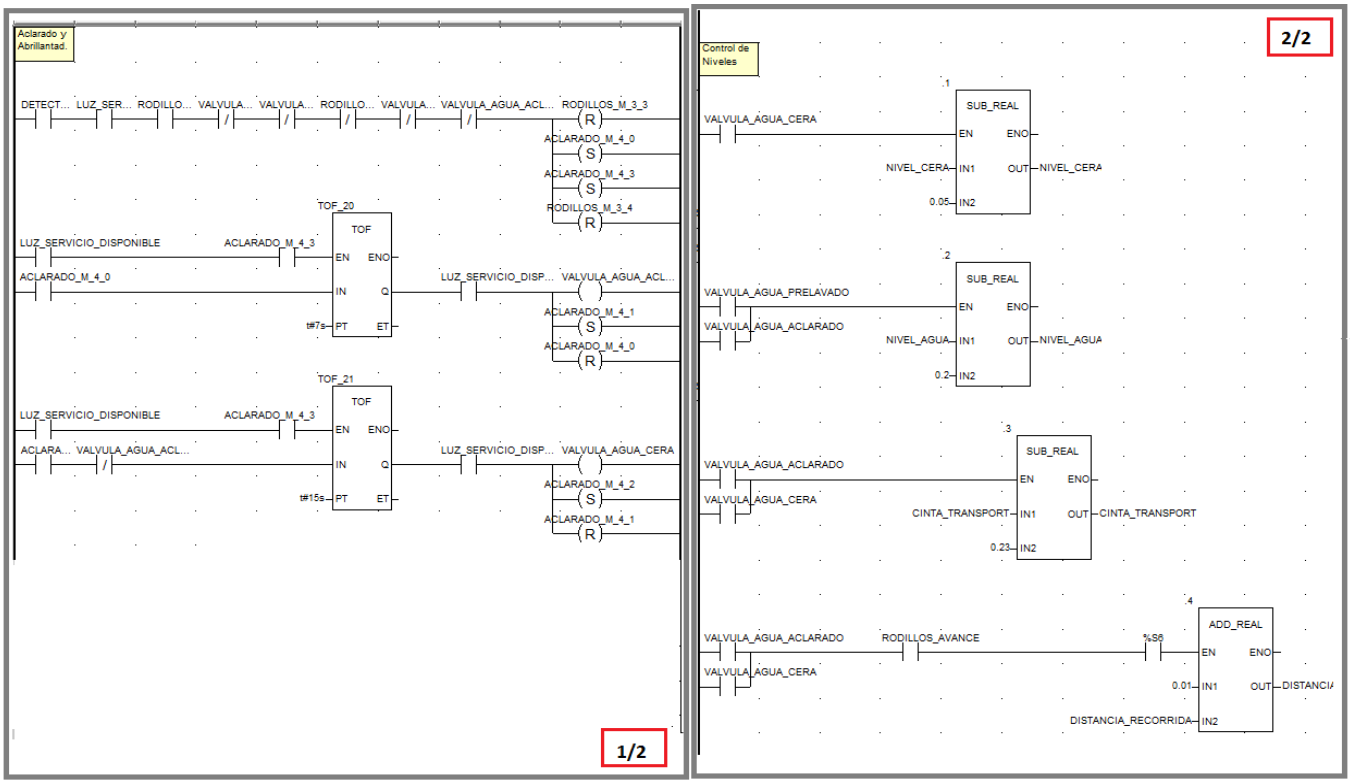


Figura 37: Código de la etapa Aclarado y Control de niveles en lenguaje LD

4.3.6 Lenguaje FBD

El diagrama de funciones (también conocido como esquema básico de funciones EBF o funtion block diagram FBD) es un lenguaje gráfico.
Los programas son bloques cableados entre sí de forma análoga al esquema de un circuito. Tiene un interface de E/S bien definida, y además y lo más importante es que posee un código interno oculto. [13]

Ventajas:

- Documentación y programación en un mismo elemento del programa
  - Informes generales, comentarios, flujos de datos…
- Aplicación universal, enteros, puntos flotantes
- Programación estructurada
  - Definición y llamada a subrutinas
- Conjunto de funciones y de bloques estandarizados
  - Se pueden mezclar bloques de distintos fabricantes
  - Se pueden definir nuevos bloques
- Los FBs son altamente reutilizables
  - En un mismo programa
  - En programas diferentes
  - En diferentes proyectos

Como decíamos anteriormente en este modo de programación podemos crearnos nuestros propios bloques que me desempeñen una función específica definida por nosotros.
A continuación veremos cómo crear nuestro DFB para simular por ejemplo un marcha/para de un motor.

El primer paso es crearnos una variable tipo DFB. Aquí le diremos cuantas entradas, salidas y la sección de programa que debe cumplir nuestro bloque:

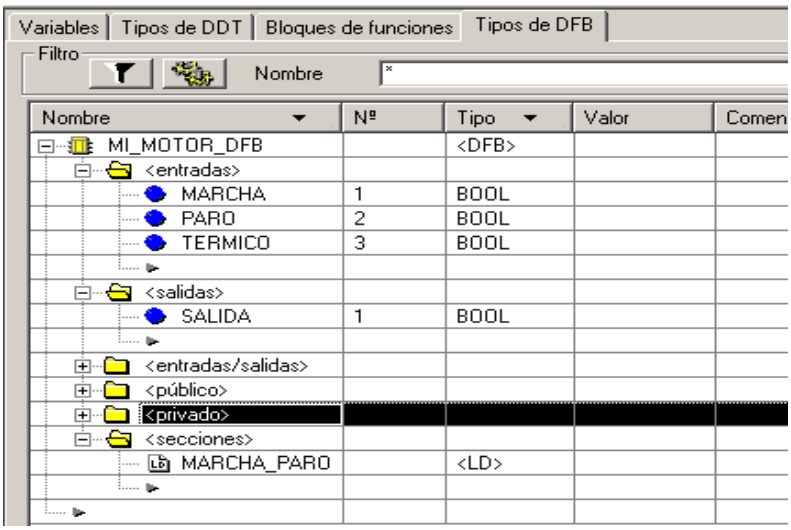


Figura 38: Creación de variable tipo DFB

Podemos diferenciar diferentes partes dentro de nuestra variable de creación del DFB:

Entradas: Entradas de las patillas del bloque.

Salidas: Salidas de las patillas del bloque.

Ent/Sal: Variables estructuradas.

Público: Variables de uso de programa.

Privado: Variables de uso exclusivo por el DFB.

Secciones: Secciones donde crearemos la función del bloque.

Dentro de la sección creada hacemos nuestro programa en el lenguaje LD:

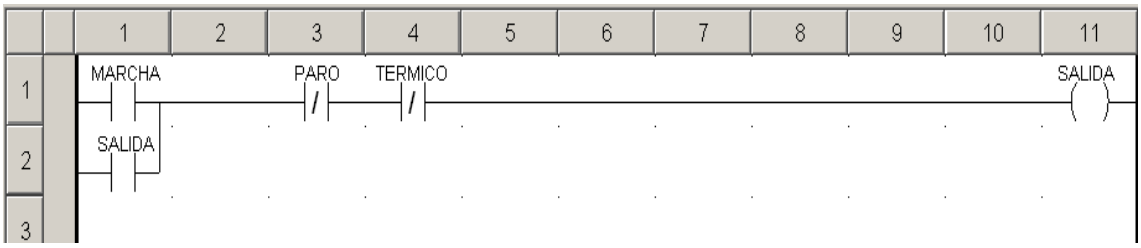


Figura 39: Variables que serán las entradas/salidas de nuestro bloque

Y con esto ya podremos crear nuestro bloque de función.

Una vez creado el bloque ya estamos listo a introducirlo en nuestra sección FDB de la forma que explicamos antes.

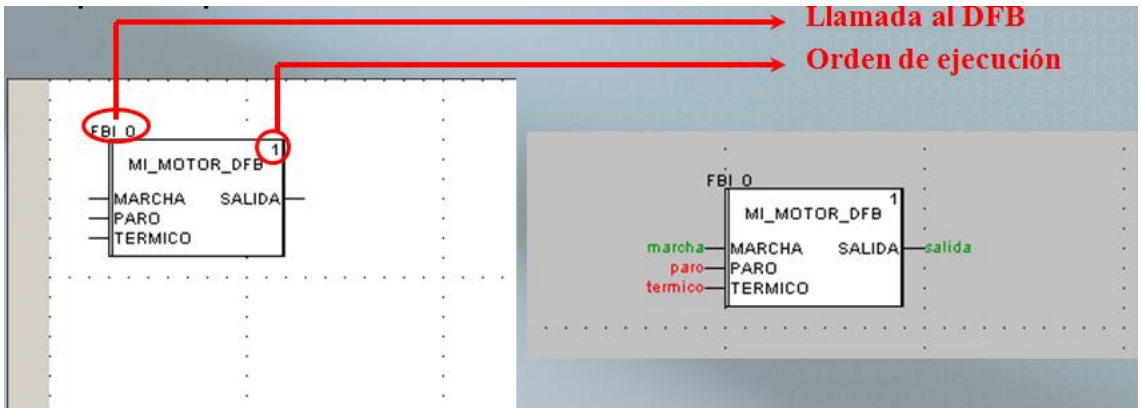


Figura 40: Bloque creado con el lenguaje interno de la figura anterior

Se diferencian los bloques ya creados con los creados por el usuario por que en los laterales del bloque tiene una doble línea.

Pues una vez explicada como se crean bloques FBD, lo aplicamos para nuestro proceso de automatización, he realizado este tipo de programación porque a la hora de simular con el PLC físico del laboratorio es más fácil asignar las entradas/salidas que vayas a utilizar. De esta

manera quedará más ordenado y se verá todo en una ventana, puesto que el código es interno y no se verá.

Nuestra estación con los bloques FBD creados quedarán como indican en la figura 41:

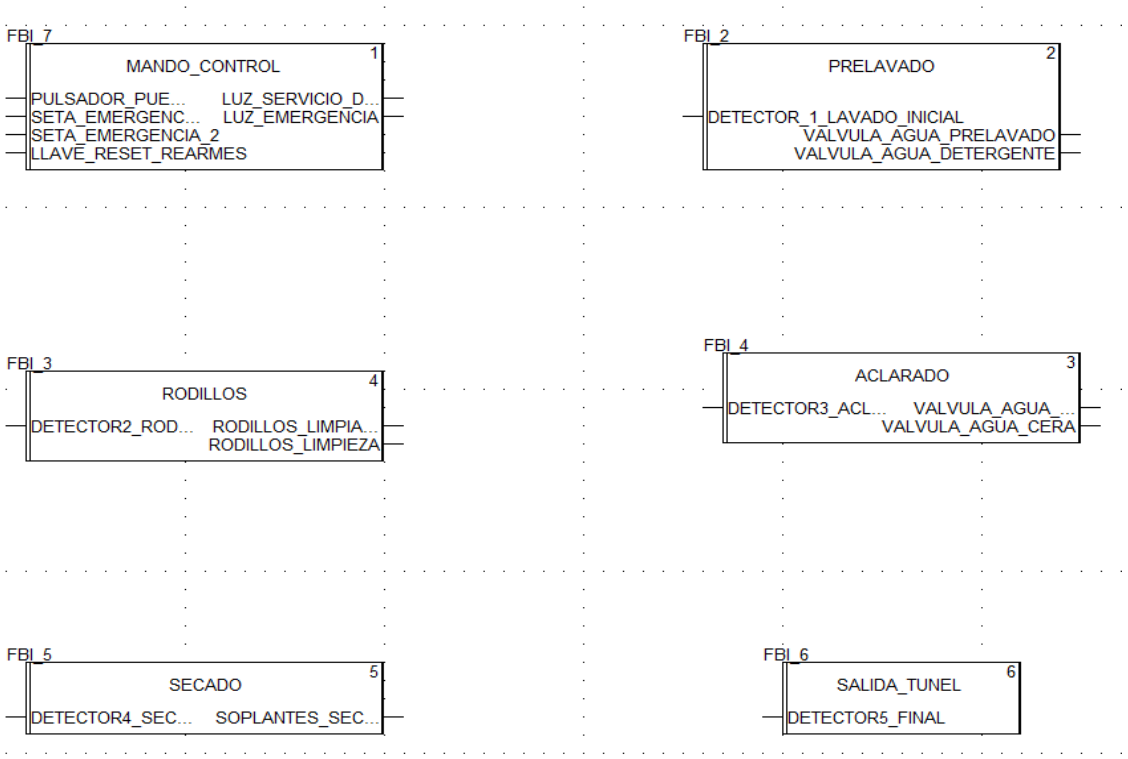


Figura 41: Bloques FBD de nuestra estación

De tal manera que si a las entradas y salidas se le asignan estas variables establecidas por el programador, el sistema funcionaría correctamente.

Entradas (Bastidor 0)	
%I0.1.0	PS (Pulsador_Servicio)
%I0.1.1	SE1 (Seta_emergencia_1)
%I0.1.2	SE2 (Seta_emergencia_2)
%I0.1.3	RR (Reset_Rearmes)
%I0.1.4	D1 (Detector_1_Inicial)
%I0.1.5	D2 (Detector_2_Rodillos)
%I0.1.6	D3 (Detector_3_Aclarado)
%I0.1.7	D4 (Detector_4_Secado)
%I0.1.8	D5 (Detector_5_Salida)

Salidas (Bastidor 0)	
%Q0.1.0	LS (Luz_de_servicio)
%Q0.1.1	LE (Luz_de_emergencia)
%Q0.1.2	RA (Rodillos_avance)
%Q0.1.3	VAP (Válvula_agua_prelavado)
%Q0.1.4	VAD (Válvula_agua_detergente)
%Q0.1.5	RLZ (Rodillos_limpieza)
%Q0.1.6	RLL (Rodillos_limpiallantas)
%Q0.1.7	VAA (Válvula_agua_aclarado)
%Q0.1.8	VAC (Válvula_agua_cera)
%Q0.1.9	SS (Soplantes_secado)

Tabla 2: Entradas/Salidas digitales del bastidor 0

5. SIMULACIÓN DEL EJEMPLO PROGRAMADO

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

5. SIMULACIÓN DEL EJEMPLO PROGRAMADO:

5.1 Pantalla de operador

Las pantallas de operador son pantallas en las que es posible insertar objetos como botones, indicadores, textos, números, barras, casilla de verificación, imágenes desde una librería de pantallas de operador o desde su disco, etc. Su diseño es totalmente libre o sea que no hay que seguir ninguna regla especial. Se utilizan las herramientas más tradicionales como copiar, cortar, pegar y el ratón para desplazar los objetos o acceder a las propiedades (mediante un doble clic).
Para que una pantalla de operador sea lograda, se requiere mucho tiempo a nivel de diseño. La librería de operador permite utilizar numerosos objetos gráficos ya creados que facilitan la creación de pantallas de operador.

5.1.1 Creación y Definición

A la hora de visualizar nuestro proceso descrito en el capítulo anterior de manera gráfica, con el software de Unity Pro tenemos la posibilidad de reproducir la simulación mediante gráficos y poder ver que ocurre cuando el proceso está en marcha a través de nuestro simulador de manera offline.

Los elementos gráficos se crean mediante "Pantalla del operador". A través de una herramienta que aporta Unity Pro, en la que el usuario puede crear pantallas, en las que es posible insertar objetos estáticos o dinámicos que pueden ser botones, indicadores, textos, números, barras o casilla de verificación. El diseño es totalmente al gusto del programador, no hay que seguir ninguna regla.

Para crear una pantalla de operador se ha de hacer un clic con el botón derecho sobre pantallas de operador desde el explorador de proyectos y seleccionar nueva pantalla, tal y como se indica en la siguiente figura 42:

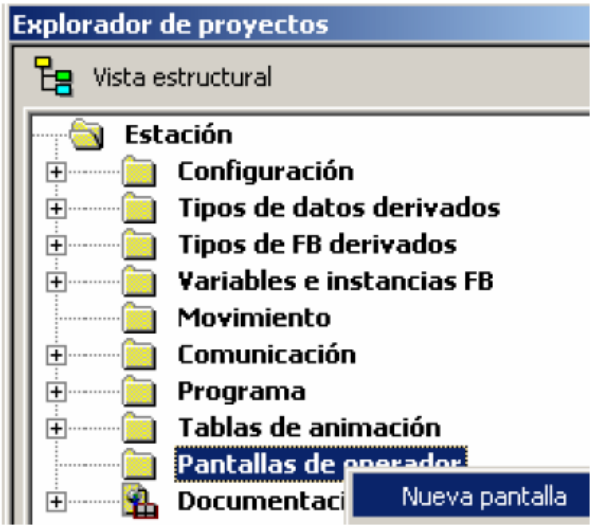


Figura 42: Creación nueva Pantalla de operador

Aparecerá una ventana de propiedades en la que es posible cambiar el nombre la pantalla. También es posible cambiar el tamaño de la pantalla en la pestaña Visualización.

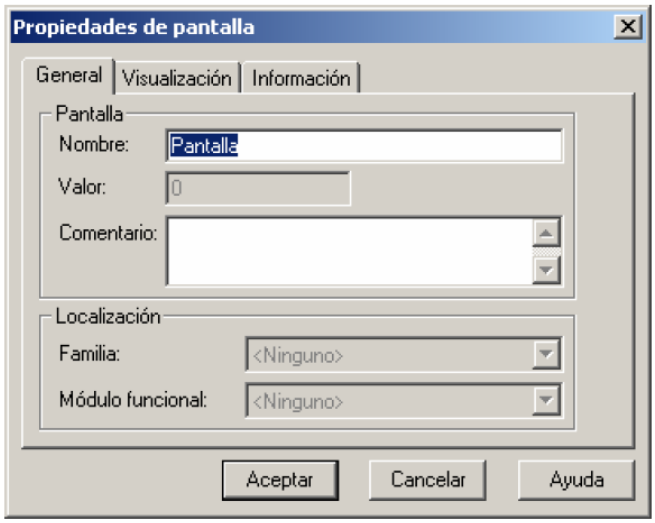


Figura 43: Propiedades de la Pantalla de Operador

De esta manera ya estaría generada nuestra pantalla de operador, ahora el siguiente paso es configurarla a nuestro gusto, en mi caso, ha quedado de esta manera:

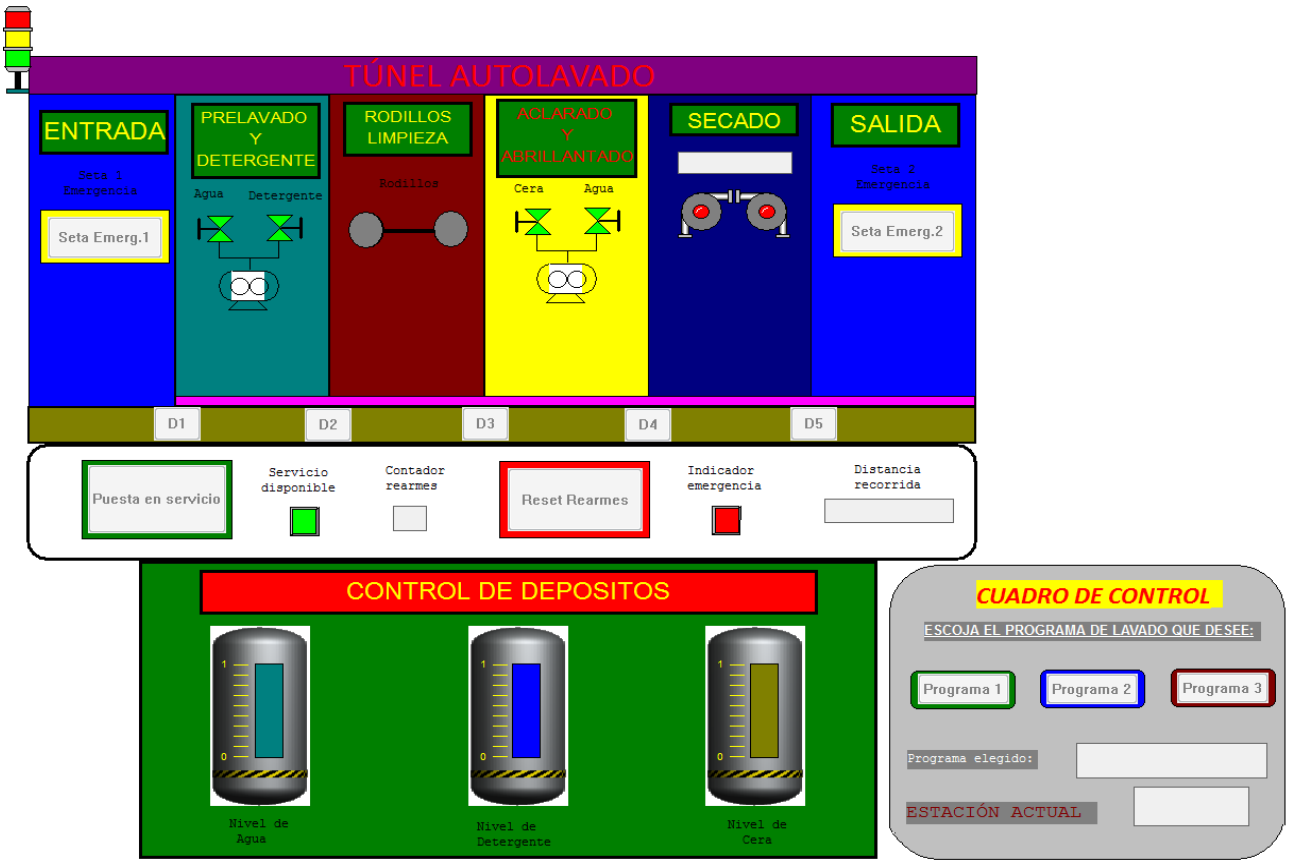


Figura 44: Pantalla de Operador de nuestra estación

5.1.2 Librería pantalla de operadores y desarrollo

Para añadir elementos animados, se puede hacer uso del conjunto de librerías que el software incorpora. Estas librerías, así como las familias, funciones y variables (estructuras de datos de E/S) se pueden utilizar para desarrollar un proyecto de automatización. También es posible añadir librerías nuevas, familias nuevas y gestionar las versiones de los bloques con la función de insertados en la librería. El acceso es mediante la barra de herramienta. Aparecerá una nueva ventana en la que se podrá desplegar conjuntos animados de distintas temáticas. En la imagen se puede observar como acceder a la Librería de pantallas de operador.

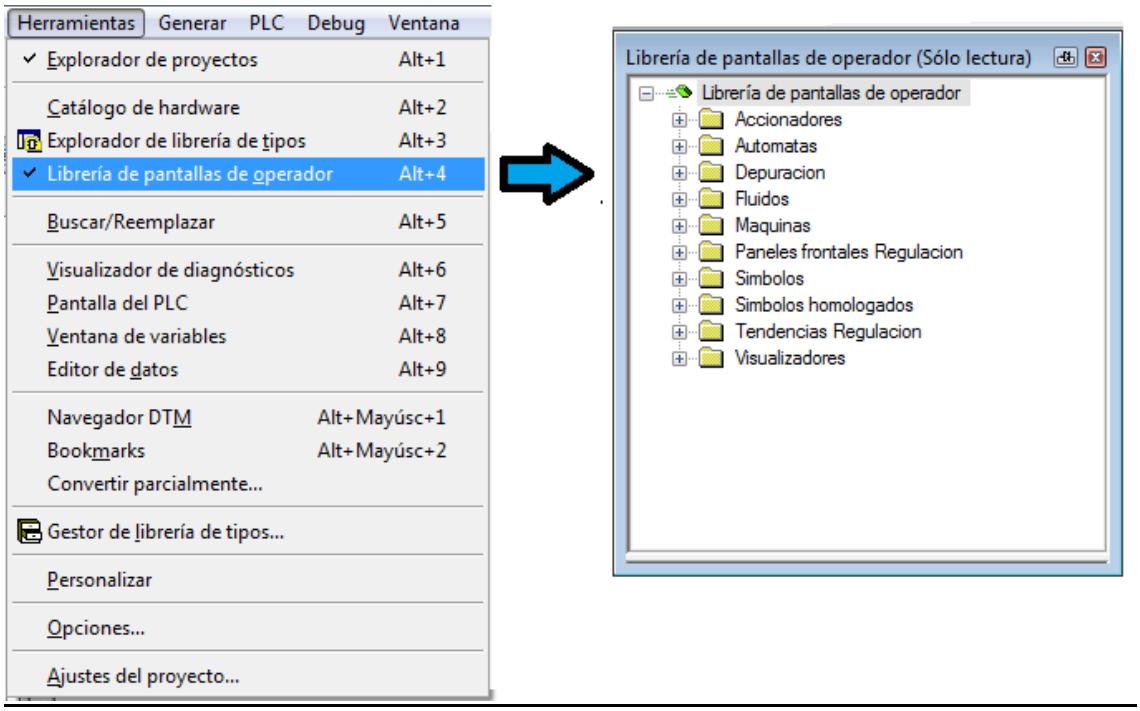
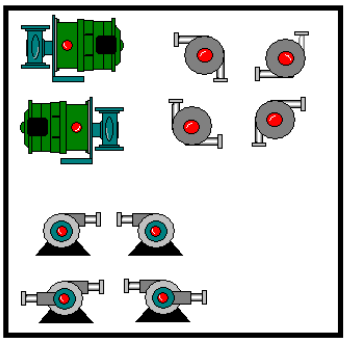


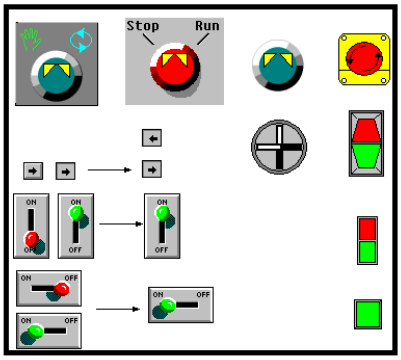
Figura 45: Acceso a las Librería de pantallas de operador

ACCIONADORES

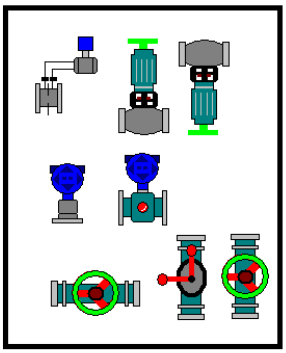
BOMBAS



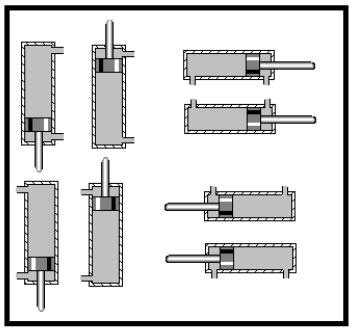
BOTÓN



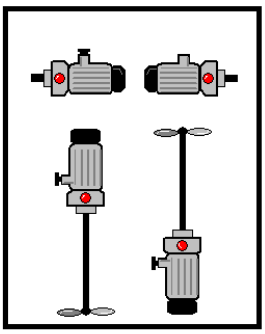
VÁLVULA



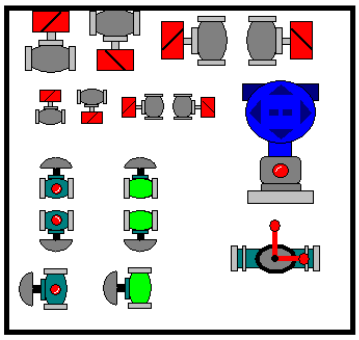
GATO



MOTOR



VÁLVULA

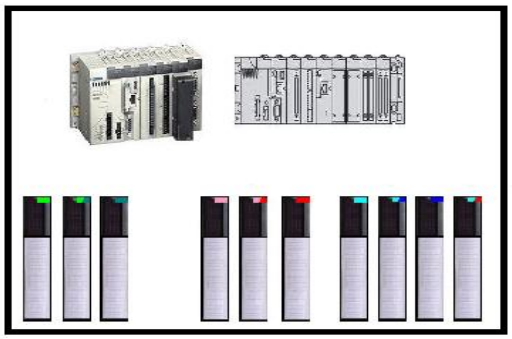


AUTÓMATAS

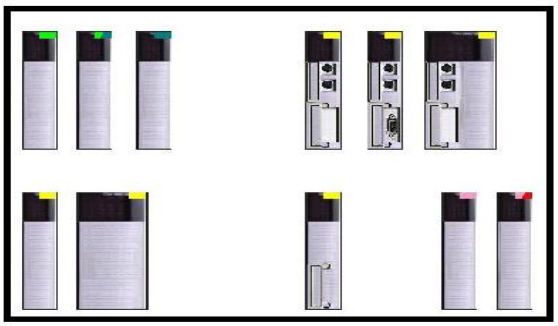
CANopen

CCX

Modico M340

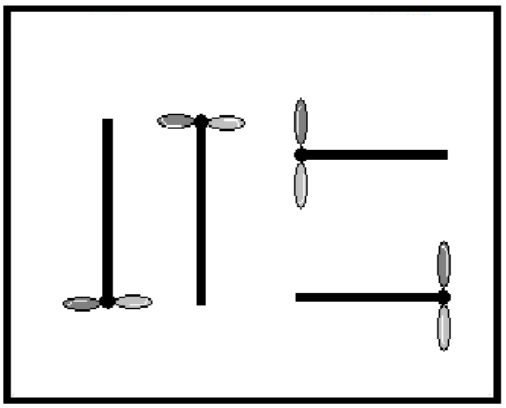


Premium

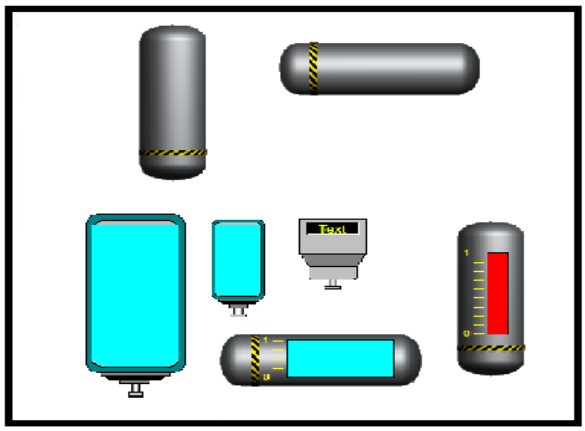


FLUIDOS

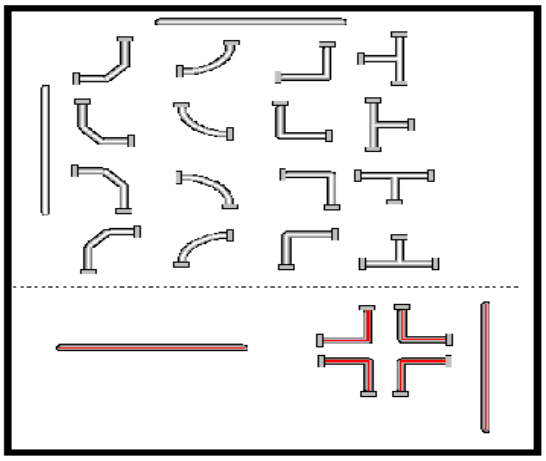
Agitador



Cuba

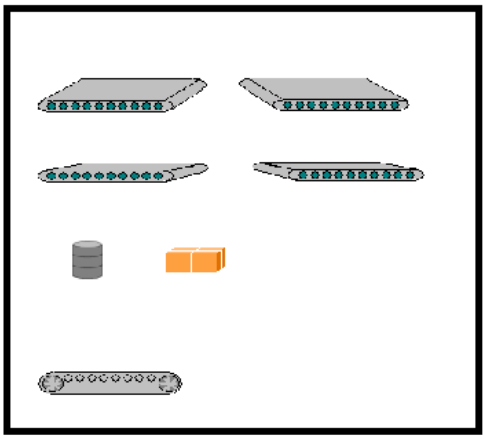


Tubo

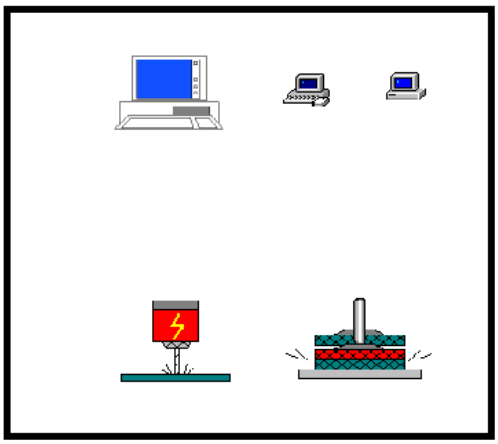


MÁQUINAS

Cinta Transportadora

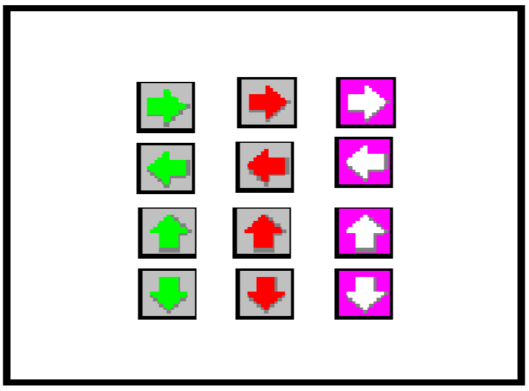


Máquina



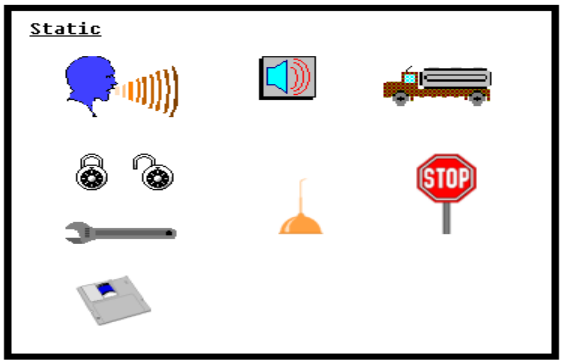
SÍMBOLOS

Flecha



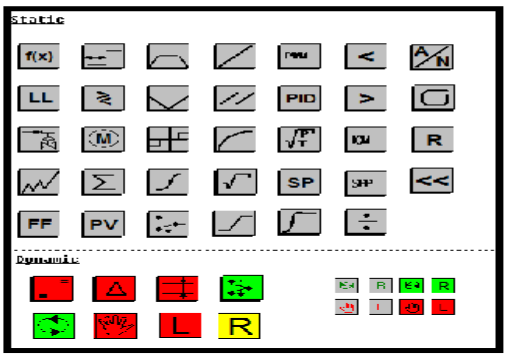
Pictograma

Varios

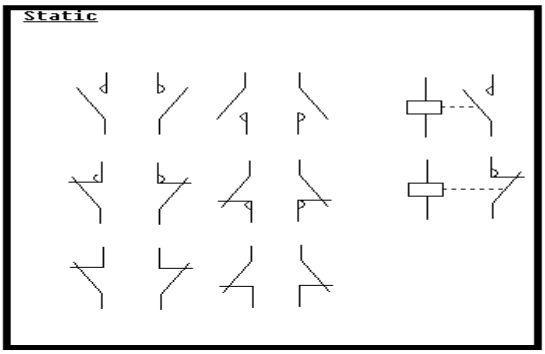


SÍMBOLOS HOMOLOGADOS

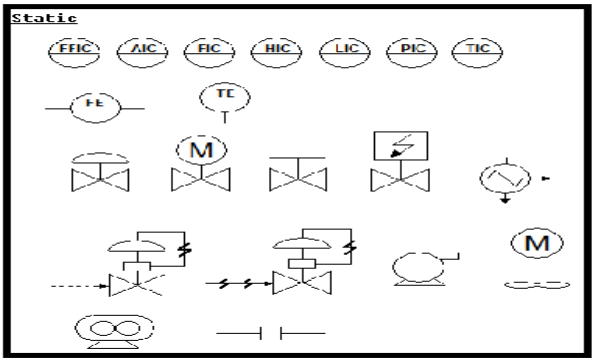
Bloque de función

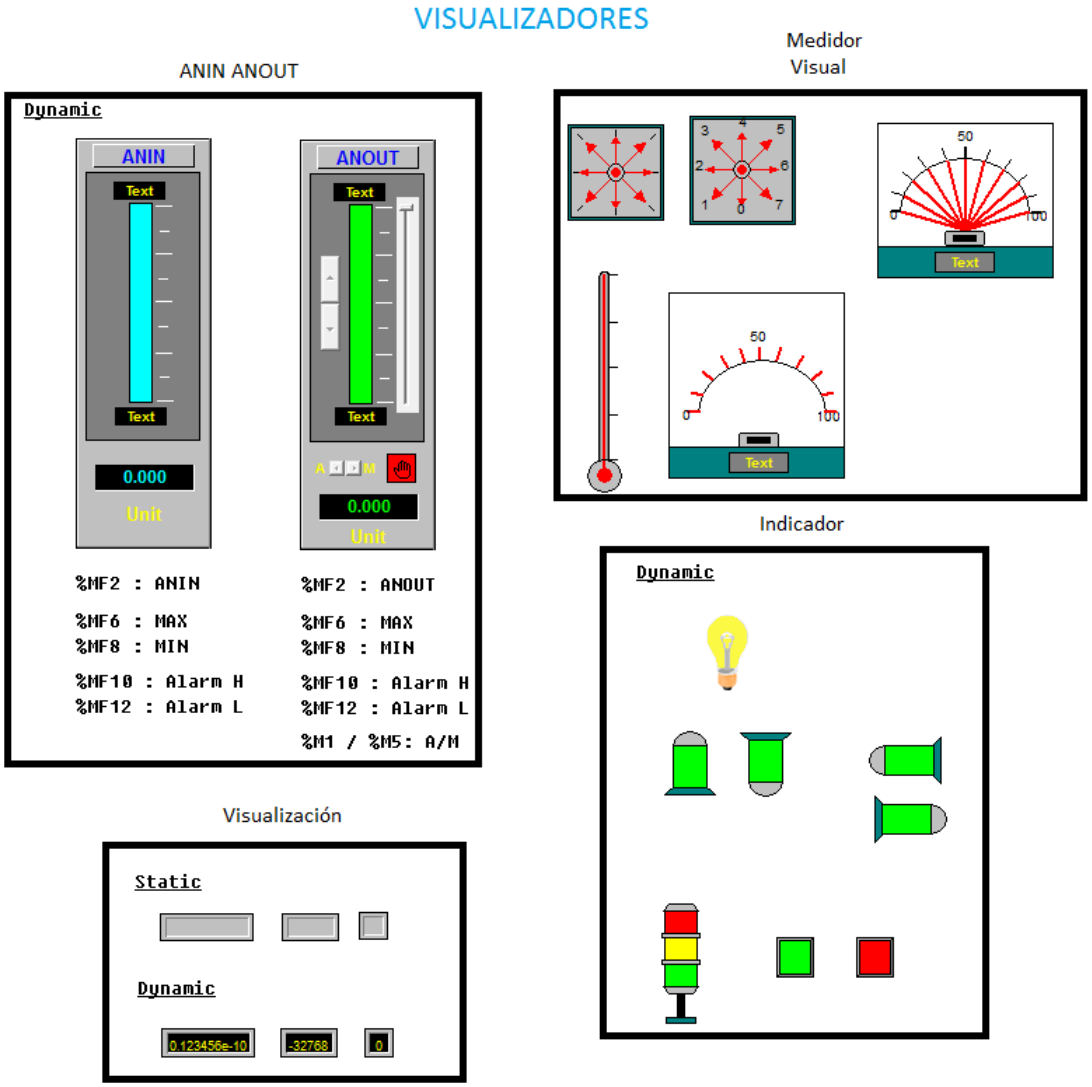


Eléctrico



Regulación





Dentro del simulador de Unity Pro podemos manipular en los gráficos del catálogo de la "librería de pantallas del operador", es decir, si queremos un agitador y por ejemplo necesitamos sólo las hélices que lo componen, podemos agrupar y desagrupar para programar cada elemento por separado, en definitiva, nos ofrece un amplio abanico para poder diseñar a nuestro gusto. Un ejemplo de agrupar y desagrupar lo podemos ver en la siguiente figura:

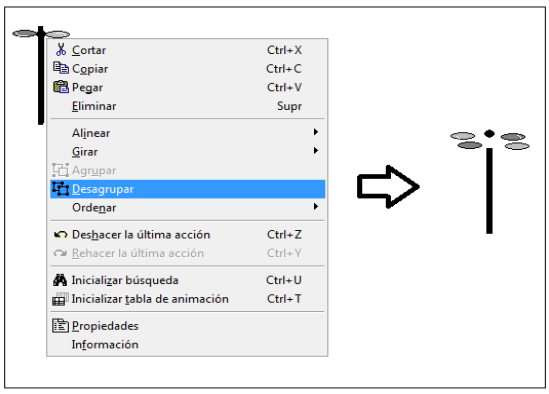


Figura 46: Desagrupar elemento agitador

Los conjuntos anteriores, están asociados a una temática que incorpora elementos estáticos y elementos dinámicos. La clasificación según el movimiento que se transmita son:

- Los ***elementos estáticos***, como su nombre indica, aparecen en la simulación representando a un objeto concreto pero no ofrece ningún movimiento en su simulación. Puede aparecer durante todo el tiempo de la duración de la simulación, o si están asociados a una variable booleana, el máximo "dinamismo" (no olvidando que se habla de elemento estático), que puede ofrecer, es que aparezca cuando esté activa la variable y desparezca cuando esté inactiva, o viceversa.
- Los ***elementos dinámicos asociados a variable booleana***, producen movimiento en el transcurso de la simulación. El dinamismo, va a ir en función del objeto.
- Los ***elementos dinámicos con variables tipo entero (INT)*** asociada, van a producir un dinamismo más real.

Es importante tener en cuenta el significado de cado uno de los colores en la simulación. Cada objeto lleva etiquetas de colores que va ha indicar si se encuentra vacío-lleno, activo-desactivo. Por tanto, el color de cada elemento, indica un estado diferente en cada momento. La Tabla 3 contiene el resumen de colores.
El color de los estados los decide el propio programador a su gusto personal, tal y como se explica en la siguiente figura.
El ejemplo de la Figura 47 explica el estado de una electroválvula donde las etiquetas utilizadas son la verde (válvula activada), la roja (válvula desactivada).

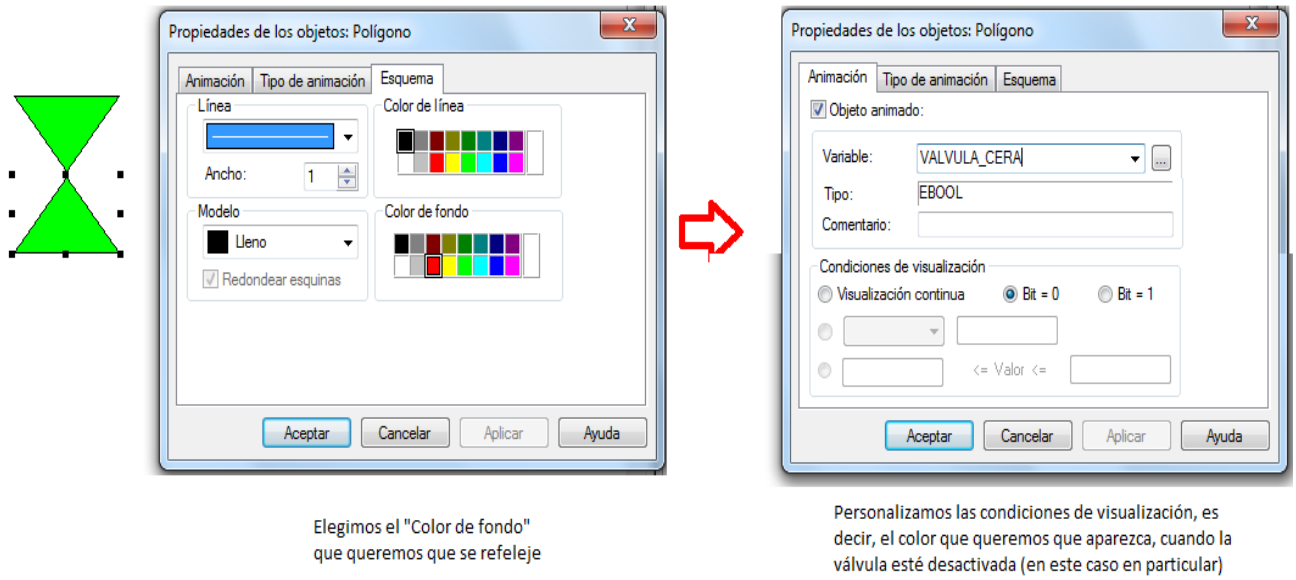


Figura 47: Personalización de objeto dinámico

En la siguiente tabla se describe cada una de las etiquetas, con su color identificativo y el objeto que afecta al mismo.

Nombre del color	Color	Etiqueta	Objeto afectado	Significado
Verde azulado		Nivel agua	Tanque de control para el nivel de agua	Estado numérico del tanque de control
Verde		Abierto, activado, ON	Válvulas, motor, Estado del servicio, Estación actual	Válvula abierta, motor encendido, estado ON, estación ON
Rojo		Cerrado, desactivado, OFF	Válvulas, motor, Estado del servicio, Estación actual	Válvula cerrada, motor apagado, estado OFF, estación OFF
Azul oscuro		Nivel detergente	Tanque de control para el nivel de detergente	Estado numérico del tanque de control
Amarillo oscuro		Nivel cera	Tanque de control para el nivel de cera	Estado numérico del tanque de control
Fucsia		Cinta transportadora	Cinta transportadora	Estado para saber en que posición se encuentra el vehículo

Tabla 3: Significado de colores en la simulación del proceso

5.1.3 Configuración del ejemplo desarrollado

Con la combinación de todos los objetos de la librería, se puede llegar a la configuración que muestra la Figura 48:

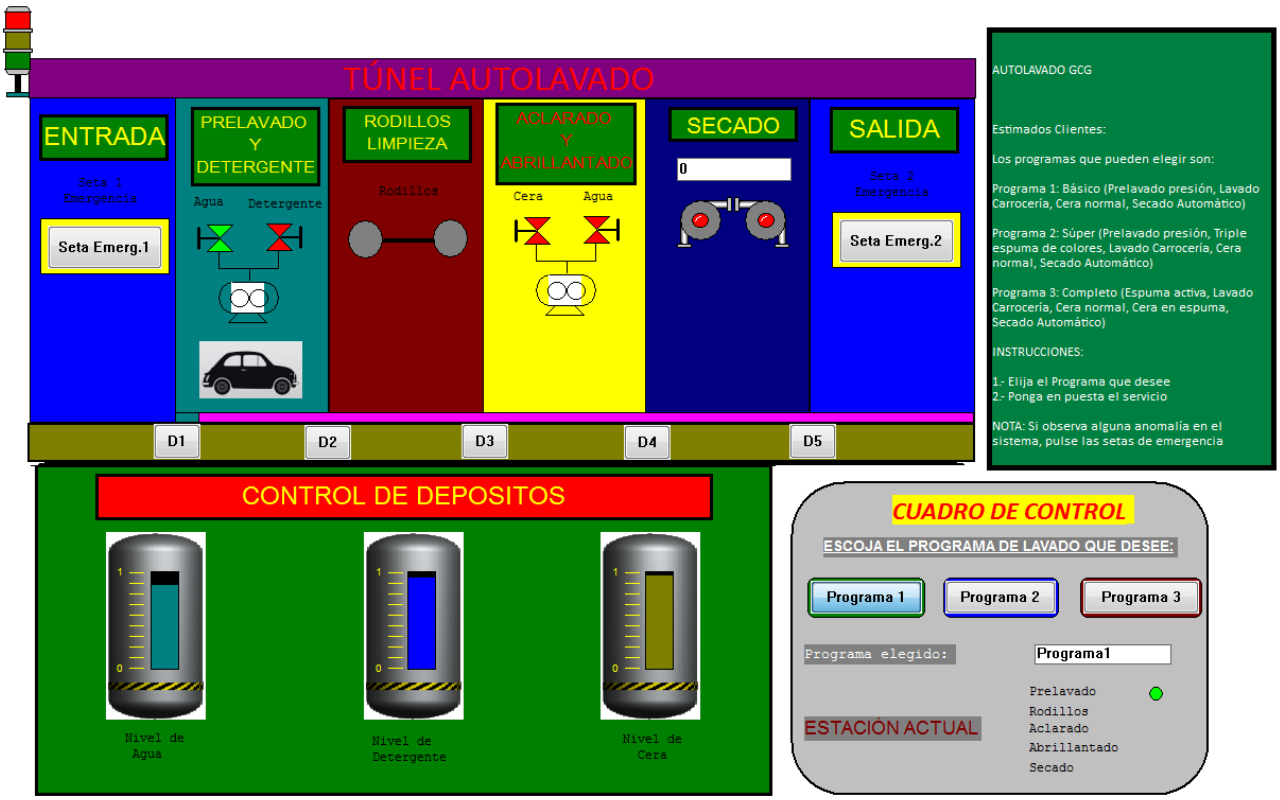


Figura 48: Pantalla de operador en funcionamiento

Cada uno de los colores va a ir cambiando modificándose a lo largo del proceso dependiendo del estado en que se encuentre el vehículo y las repeticiones que haya realizado, indicando un estado diferente de cada uno de los objetos. La Figura intenta mostrar cómo se puede crear un sistema en dos dimensiones de una estación real. Es decir, las posibilidades que ofrece Unity Pro en la combinación de objetos para representar un sistema completo de tres dimensiones. No hay que olvidar que la Figura es una figura estática plana. El sentido de los colores aparece durante la simulación, cuando el sistema está en proceso.
Para proceder a la explicación de cómo se han creado cada una de las etiquetas comentadas en el dibujo de la Figura, se va a ir analizando los objetos por separado:

Sensores de etapa, pulsador de puesta en servicio, setas de emergencia:

Dependiendo del programa que utilice el usuario, la estación funcionará de una manera u de otra.

Los programas que pueden elegir son:

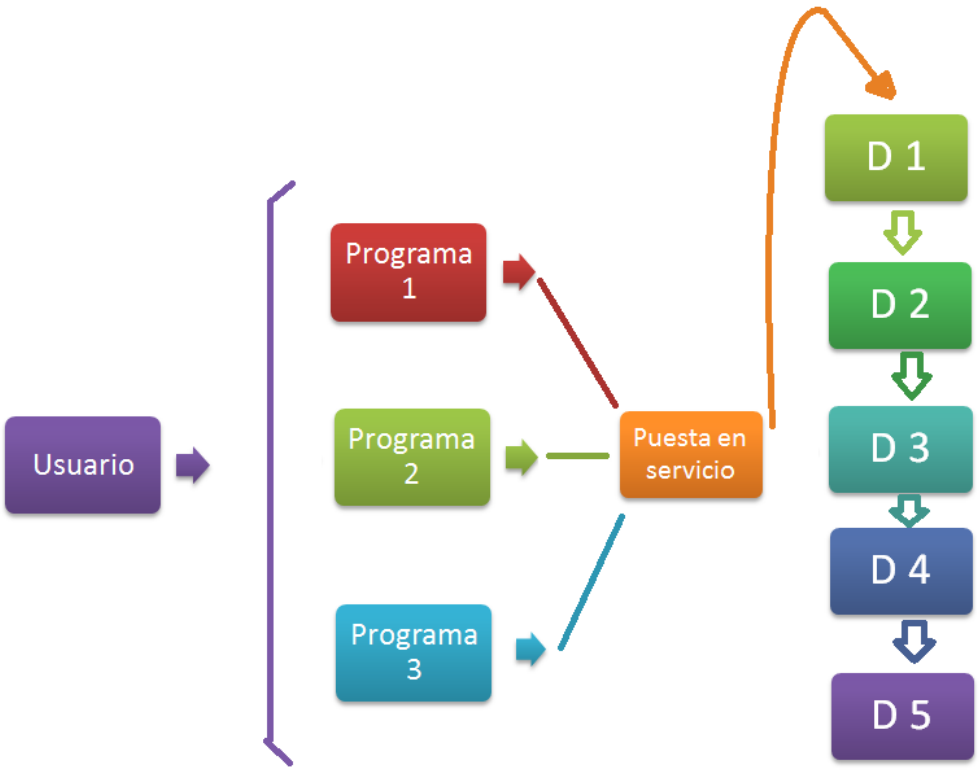
Programa 1: Básico (Prelavado presión, Lavado Carrocería, Cera normal, Secado Automático).

Programa 2: Súper (Prelavado presión, Triple espuma de colores, Lavado Carrocería, Cera normal, Secado Automático).

Programa 3: Completo (Espuma activa, Lavado Carrocería, Cera normal, Cera en espuma, Secado Automático).

Al tener cada programa una configuración diferente de lavado, lo que se ha optado para que no fuese una situación compleja, ha sido aumentar/reducir los tiempos de ciclo de cada etapa, es decir, en los programas donde se requiera más cera, aumentar el tiempo de cera, etc.

En la siguiente figura se puede observar el funcionamiento de la estación:



Si se elige el programa 1:

El programa realizará la secuencia de la manera más rápida y sencilla, tendrá los tiempos establecidos por defecto al principio del capitulo. Pulsando sobre el botón del Programa 1 y el pulsador de puesta en servicio, el coche se colocará en la estación de entrada al túnel, y una vez vaya detectando los sensores (que son los pulsadores D1, D2, D3, D4, D5; que los tenemos que simular de esta manera para que sea lo más real posible) el vehículo circule de manera correcta sobre la estación de servicio. En el programa elegido se podrá visualizar a través del cuadro de control y se regirá a los tiempos que están impuestos.

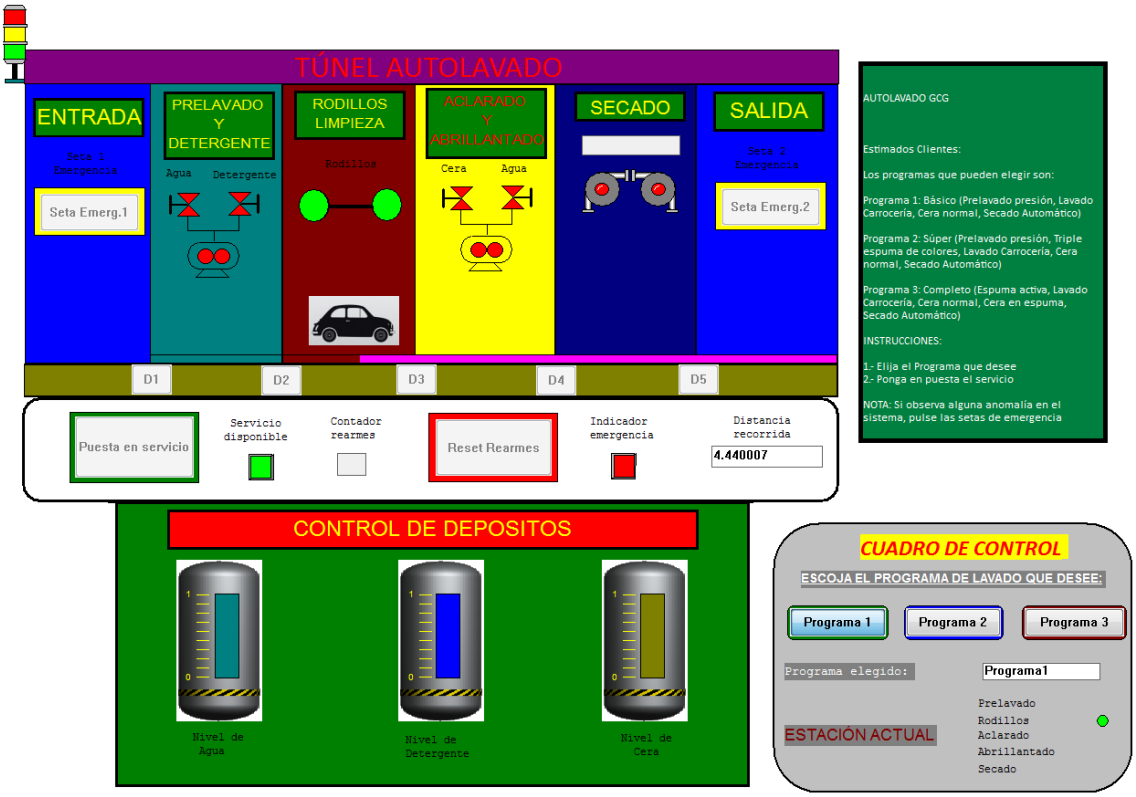


Figura 49: Programa 1 en funcionamiento

Si se elige el programa 2:

El programa realizará la secuencia de la manera mejor que la anterior, ampliando los tiempos de apertura de válvulas de agua y detergente, consiguiendo una mejor limpieza del vehículo. Pulsando sobre el botón del Programa 2 y el pulsador de puesta en servicio, el coche se colocará en la estación de entrada al túnel, y una vez vaya detectando los sensores (que son los pulsadores D1, D2, D3, D4, D5; que los tenemos que simular de esta manera para que sea lo más real posible) el vehículo circule de manera correcta sobre la estación de servicio. En el programa elegido se podrá visualizar a través del cuadro de control y se regirá a los tiempos que están impuestos.

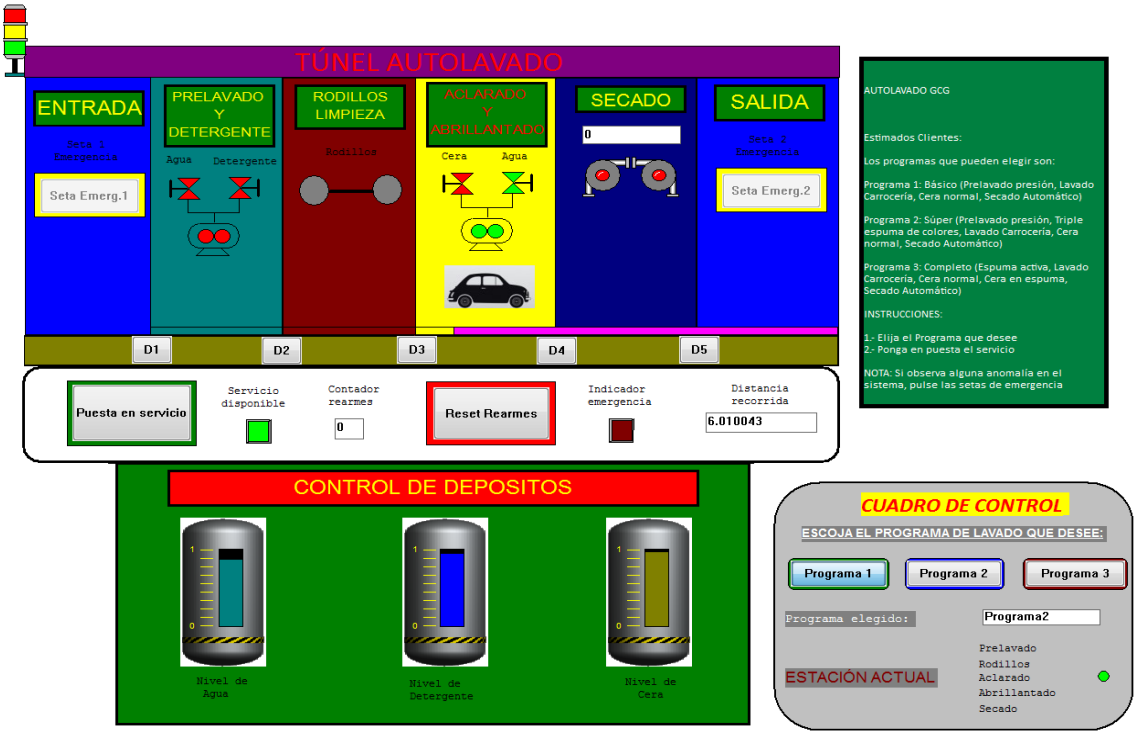


Figura 50: Programa 2 en funcionamiento

Si se elige el programa 3:

El programa realizará la secuencia de manera completa, el tiempo empleado es el máximo, ya que es la estación más larga, consiguiendo una limpieza integral del vehículo. Pulsando sobre el botón del Programa 3 y el pulsador de puesta en servicio, el coche se colocará en la estación de entrada al túnel, y una vez vaya detectando los sensores (que son los pulsadores D1, D2, D3, D4, D5; que los tenemos que simular de esta manera para que sea lo más real posible) el vehículo circule de manera correcta sobre la estación de servicio.

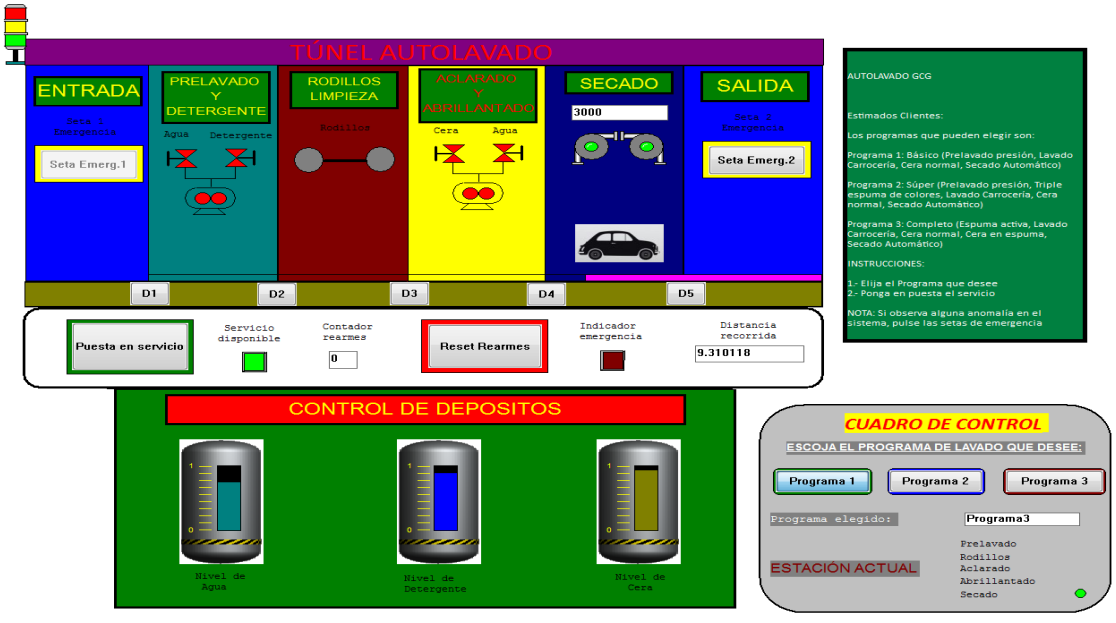


Figura 51: Programa 3 en funcionamiento

Control de Depósitos:

Cuando las válvulas de agua, detergente y cera se activan, automáticamente internamente a través de la programación, los depósitos que contienen los componentes mencionados se van vaciando.

La manera de programar estas etiquetas con las variables es cómo se explica a continuación:

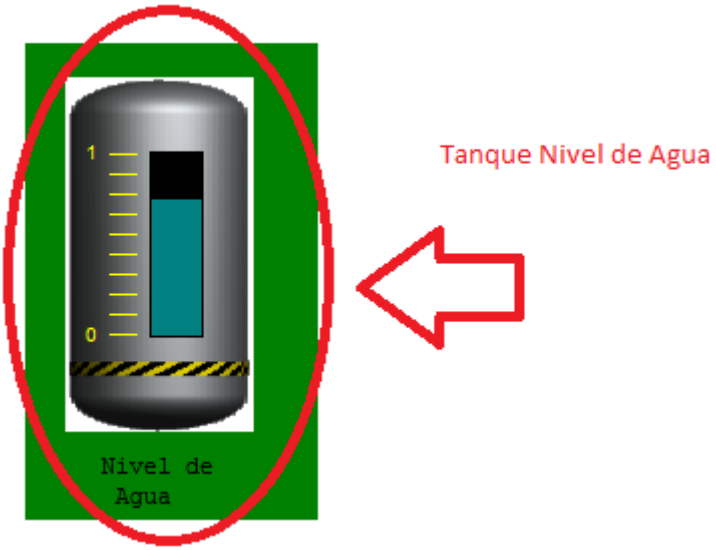


Figura 52: Programación de etiquetas

Haciendo doble clic sobre uno de los objetos dinámicos como son las tanques de control, aparece el cuadro mostrado en la Figura 52.

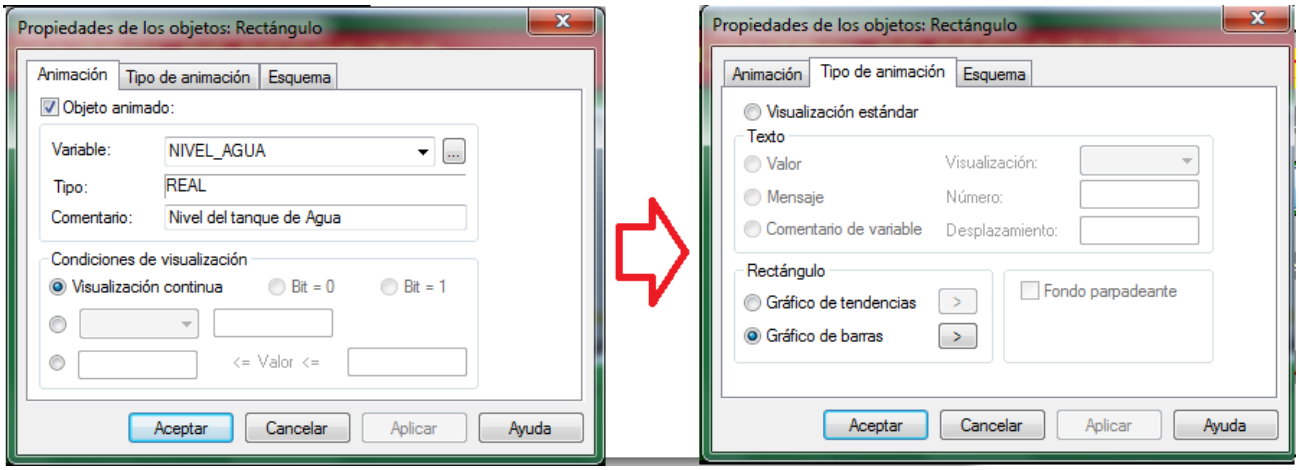


Figura 53: Elección de la variable deseada y del tipo de animación

En la siguiente pestaña, *"Tipo de animación"* se elige la función "Gráfico de barras", y pulsando sobre la flecha aparece la imagen de la derecha. En ella, se decide *"Tipo de gráfico de barras", los "umbrales", los "valores y colores predeterminados",* como muestra la Figura 54*:*

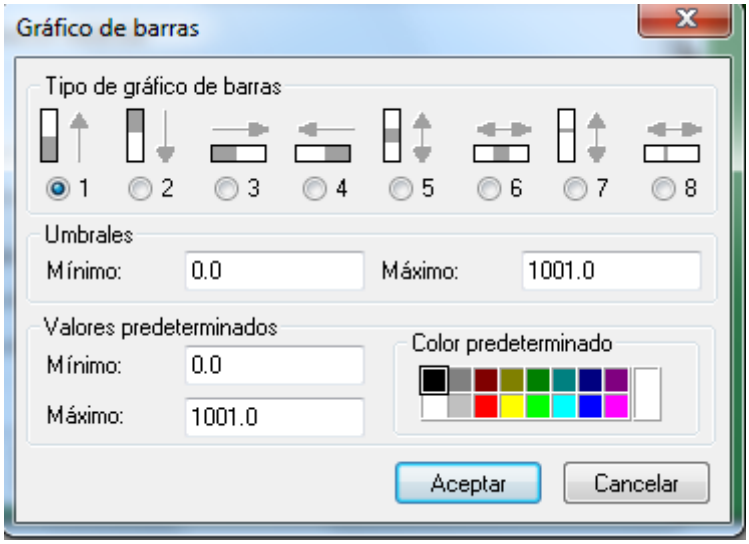


Figura 54: Configuración del gráfico de barras

Por otra parte, el control del nivel del tanque, se lleva a cabo a través de una alarma que te indica que el nivel del depósito está bajo y necesita rellenarse.

Para llevar a cabo esta operación primero hemos programado a través de una comparación del nivel del tanque y la cantidad que quieras que salte la alarma.

Figura 55: Alarmas del control de tanques

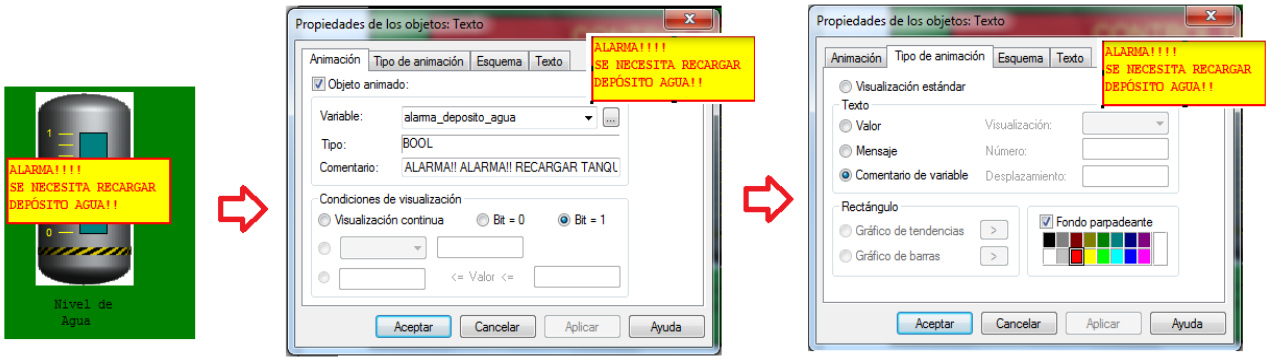


Figura 56: Configuración de etiqueta parpadeante con su respectivo mensaje

Válvulas:

Cada una de las válvulas, "V1", "V2", "V3", "V4" indica mediante color rojo que se encuentran cerradas, o mediante color verde claro que se encuentran abiertas. Estas variables han sido creadas tanto para la programación como para la simulación, y su aspecto se observa en la Figura 57:

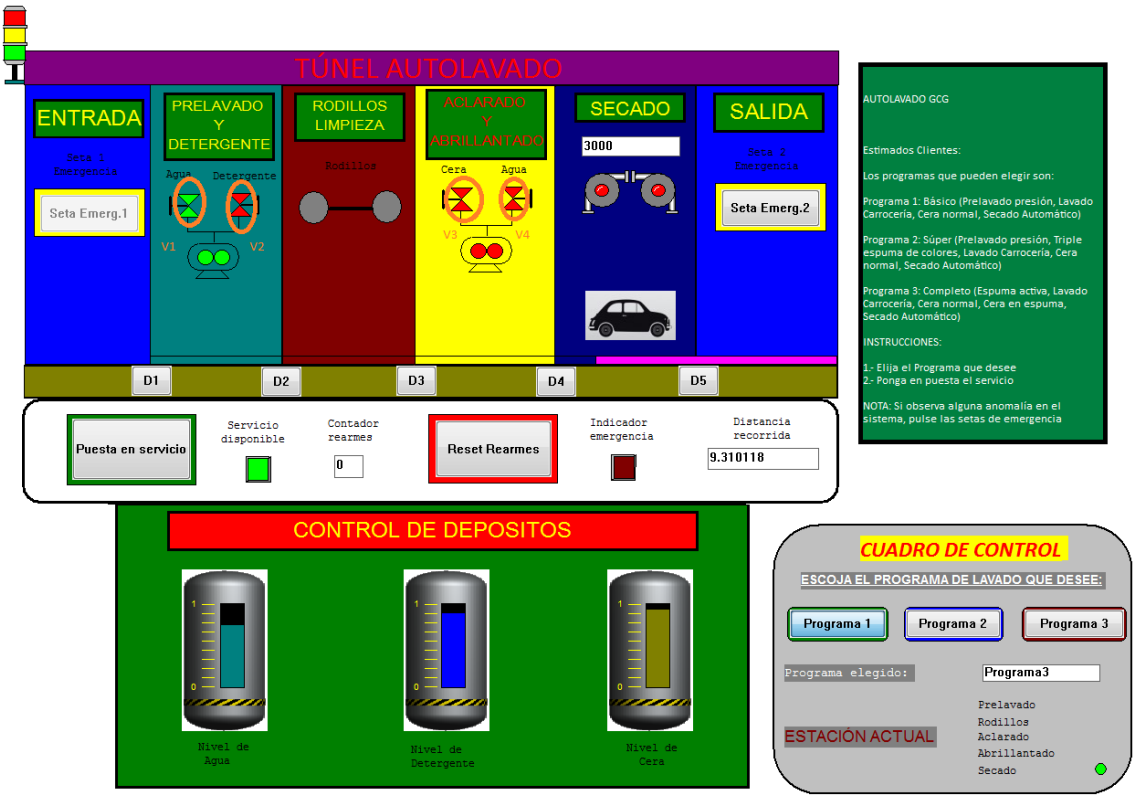


Figura 57: Válvulas utilizadas en la simulación

En la siguiente figura podemos ver el cambio de color de la Válvula 3 "V3", según su estado sea activado o desactivado:

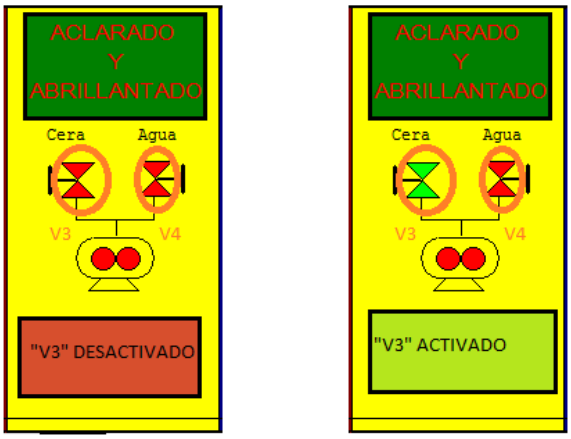


Figura 58: Estado de las válvulas

Para llevar a cabo su programación, al tratarse de variables tipo EBOOL, se han seguido los pasos mostrados en los pasos anteriores.

Motores:

Se ha creado una variable "M" asignada para el motor que se encarga del secado, es creada para la programación por ser una entrada del sistema, y es válida para la simulación. Es una variable asociada a objeto dinámico de tipo EBOOL, como las válvulas.
El motor está programado para que cuando la variable "M" esté activada se ponga en verde claro el punto central del elemento motor y las velocidades del mismo aparecen en el lector de revoluciones rpm. El objeto motor "M" se observa en la Figura 59:

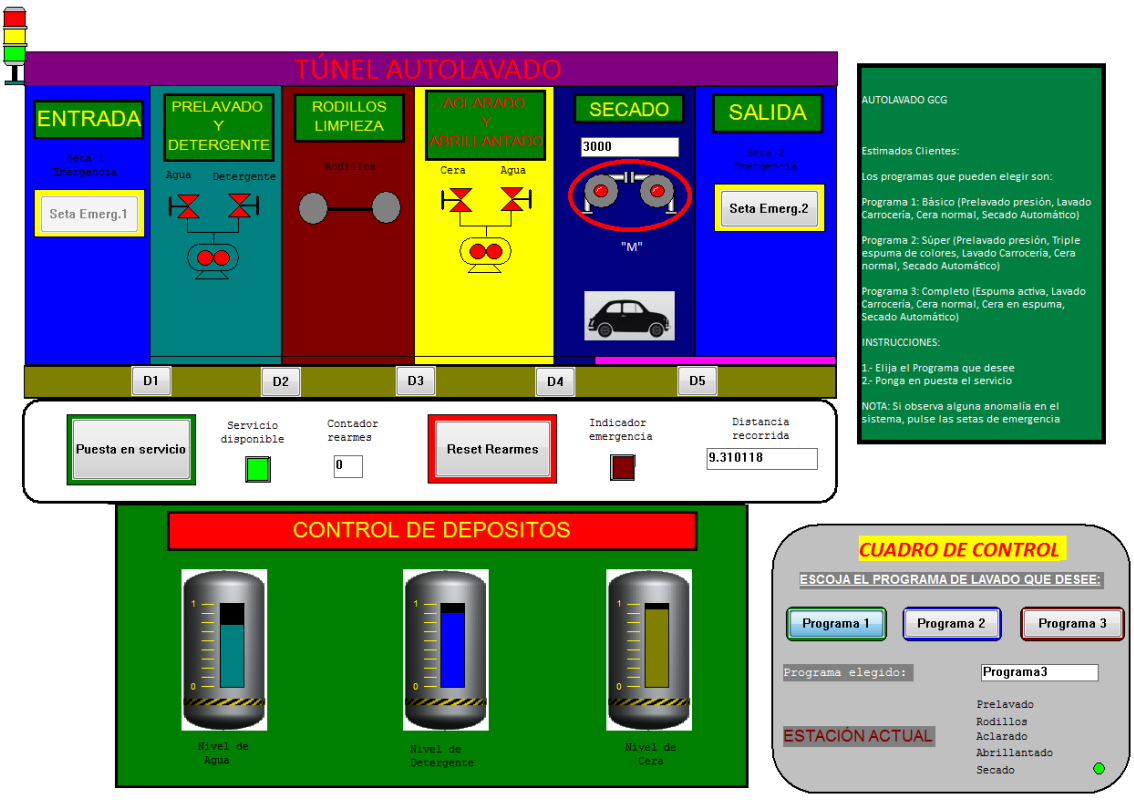


Figura 59: Motores utilizados en la simulación

La Figura 60, muestra el cambio de la variable Motor “M”, según su estado sea en marcha o parado

Figura 60: Estado de los motores

Visualizadores y Lectores de int y string:

En la siguiente Figura, se puede observar la creación de visualizadores de estado, para poder saber en todo momento si la estación se encuentra en estado disponible o hay alguna avería y por tanto el indicador de emergencia está activado. También para visualizar los rearmes producidos, la distancia recorrida, el programa elegido y las revoluciones por minuto del motor de secado hemos utilizado etiquetas de entrada, que explicaremos a continuación como se crean.

Figura 61: Campos de entrada -> Visualizadores

Para crear un campo de entrada, a través del IOS Editor:

Figura 62: Creación de un campo de entrada

Y creamos la dimensión del campo de entrada al gusto del programador. Una vez creada, haciendo doble click sobre el mismo accedemos a sus propiedades, donde vamos asignar la variable que queremos que represente al campo de entrada y por tanto que refleje el resultado.

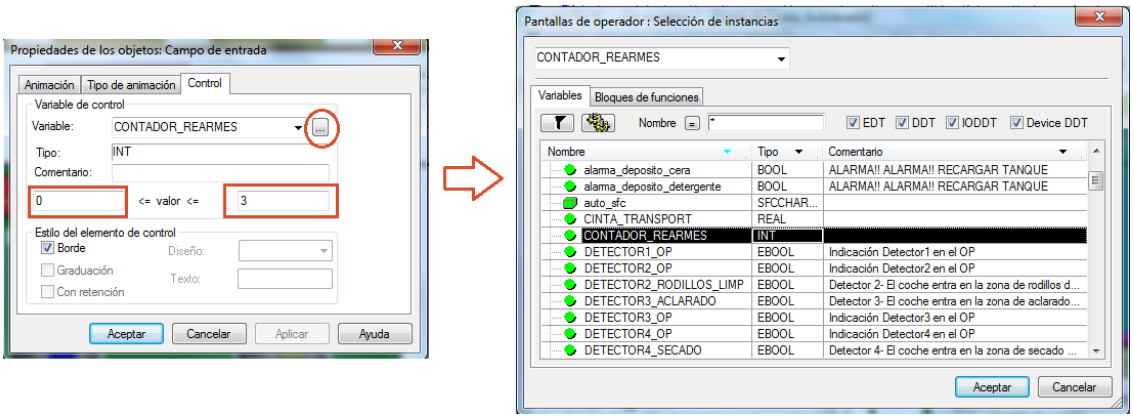


Figura 63: Propiedades y asignación de variable a los campos de entrada

Muy importante es el tipo de variable que elijas, puesto que si es un string, no podrá contener valores mínimos y máximos, sería una incongruencia.

5.2 Compilación y transferencia del Proyecto en modo de simulación

Una vez que se ha programado y se ha llegado a una solución para el problema planteado, y la pantalla de operador ha sido creada a tu gusto, es el momento de compilar el sistema para comprobar que no existan errores al compilar y por consiguiente podamos habilitar la transferencia en el simulador.
De modo que seguimos los siguientes pasos, seleccionar "*Generar*: generar todo el proyecto" corresponde al análisis y la generación del código ejecutable, como muestra la Figura 64

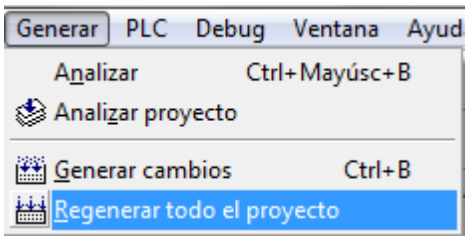


Figura 64: Regenerar proyecto

En el caso de tener errores, aparecerá la descripción de los errores en la ventana de resultados y haciendo un doble clic sobre la línea, el programa llevará directamente a la parte del proyecto que contiene el error.

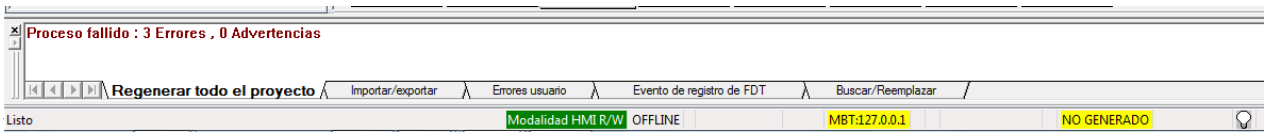


Figura 65: Ventana de errores y advertencias de compilación

Si el proyecto nos ha compilado de manera correcta, es decir, que no hay errores, a través de la pestaña "PLC" señalaremos "Modalidad de Simulación", y así activaremos el simulador del software.

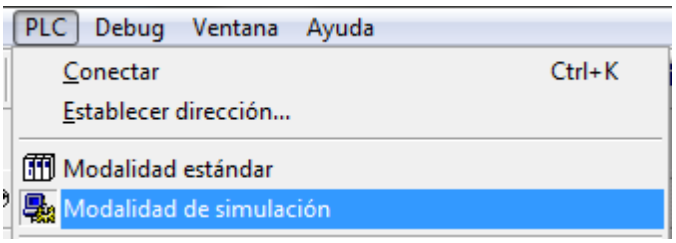


Figura 66: Conectar PLC a la modalidad de simulación

Para que funcione correctamente en esta modalidad, tenemos que configurarlo como indica en la siguiente figura:

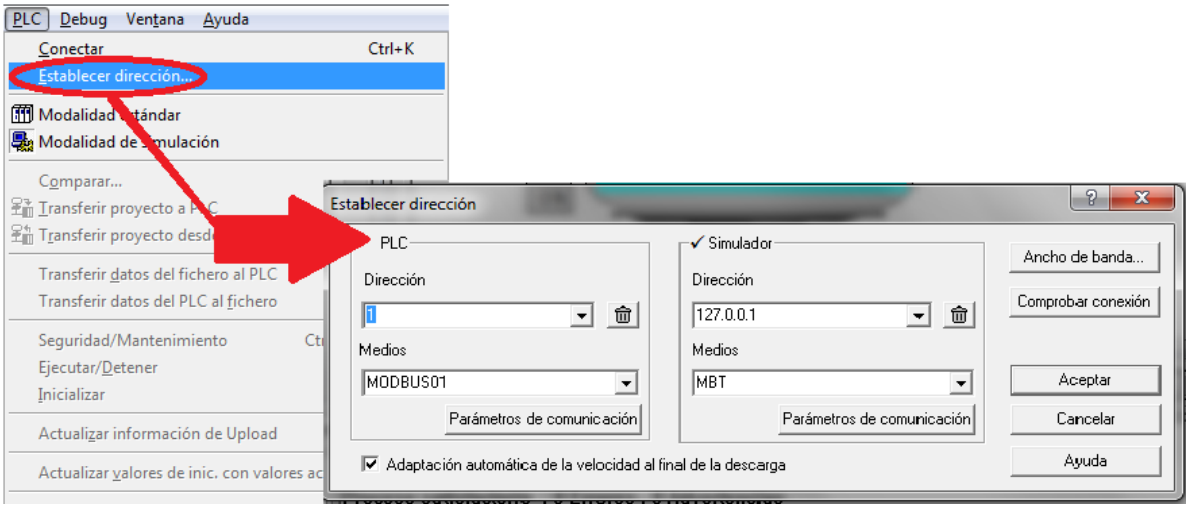


Figura 67: Establecer dirección

Si quisiéramos que la modalidad de funcionamiento sea de "Modalidad estándar", es decir, que nos conectaríamos a través del PC al autómata físico y viceversa, la configuración sería de la siguiente manera:

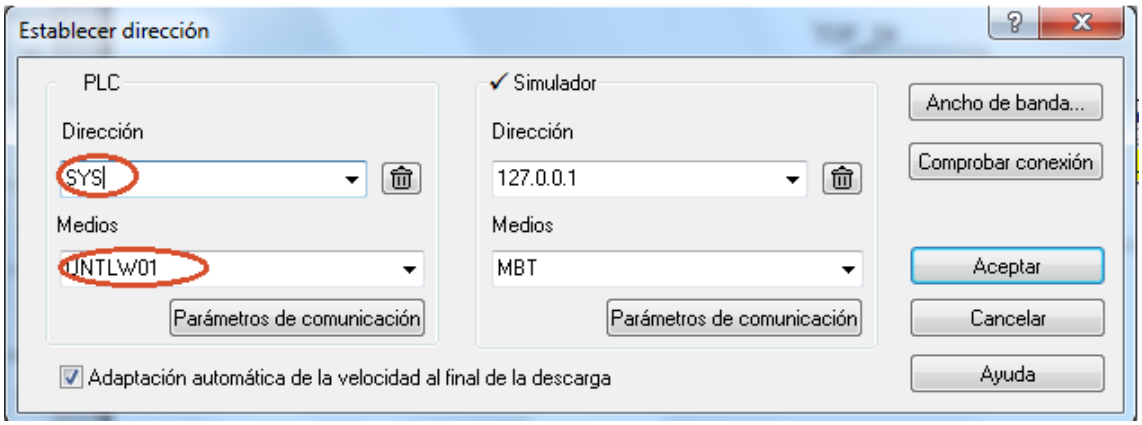


Figura 68: Modalidad estándar

Para transmitir toda la programación al propio autómata, bien esté en modalidad de simulador como modalidad estándar, primero debemos conectarnos al propio "PLC" y después "Transferir proyecto a PLC"

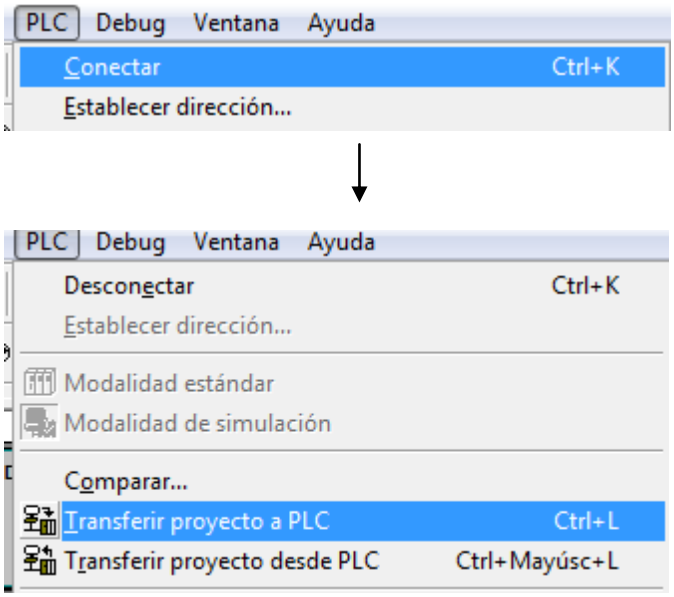


Figura 69: Transferir proyecto a PLC

El software de Unity Pro estará preparado para funcionar con el simulador mediante la barra de herramienta dando a PLC/ Ejecutar, o directamente al RUN (símbolo del play).
De esta manera el simulador estará arrancado.

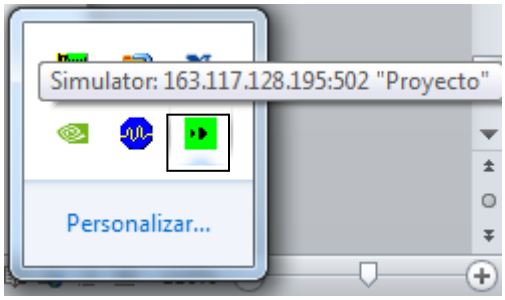


Figura 70: Simulador minimizado

Haciendo doble click en el icono verde, podemos ver la situación del panel del simulador:

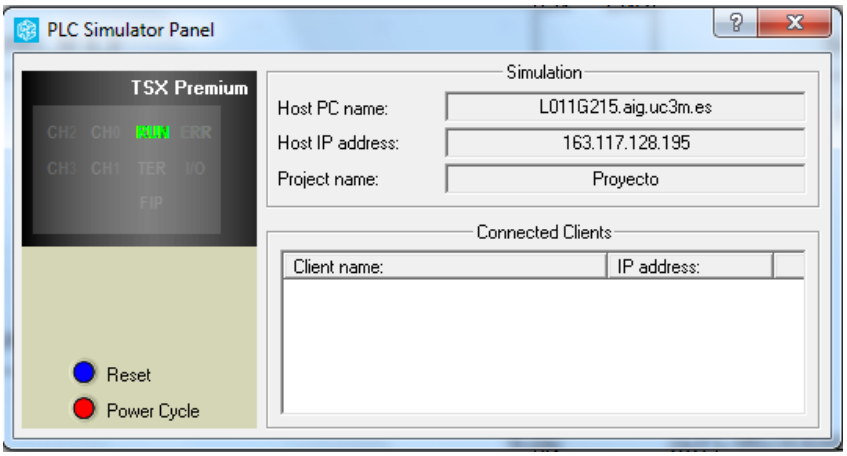


Figura 71: Simulador Unity Pro

Dicho panel del simulador nos informa en todo momento de la situación del autómata, como si se tratase de un PLC físico, el cual podemos actuar los botones de Power y reset, ver si está en RUN o existe algún error ERR.

5.3 Tablas de animación

Otra de las maneras, para que el proceso avance correctamente es mediante las Tablas de animación, las cuales, son ventanas que se crean en el explorador de proyectos y nos permiten forzar las variables dependiendo de si son digitales, analógicas o numéricas.
Si es a través de la "tabla de animación", como se desea modificar el valor de una variable, hay que hacerlo mediante el forzado de las señales con direccionamiento, con las entradas %I.
Hay que tener presente que una tabla de animación **siempre** facilitará el trabajo para el seguimiento de señales en sus cambios de valor. Pero si además, la evolución del sistema se realiza con "Tabla de animación", es de doble utilidad pues es a través de ella es donde se pueden controlar dichos cambios.
Este método es más tradicional y algo más largo a la hora de desarrollarlo, puesto que hay que volver a declarar las variables que quieres ver su evolución.
Tanto las entradas como las salidas deben estar definidas previamente en la tabla de variables.

Comenzamos creando nuestra nueva tabla de animación:

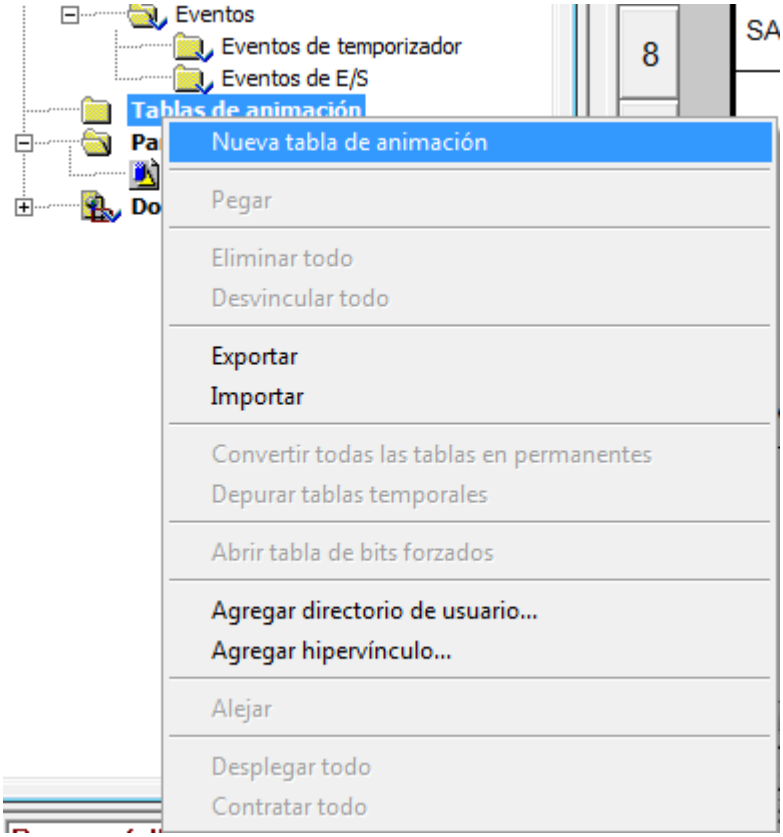
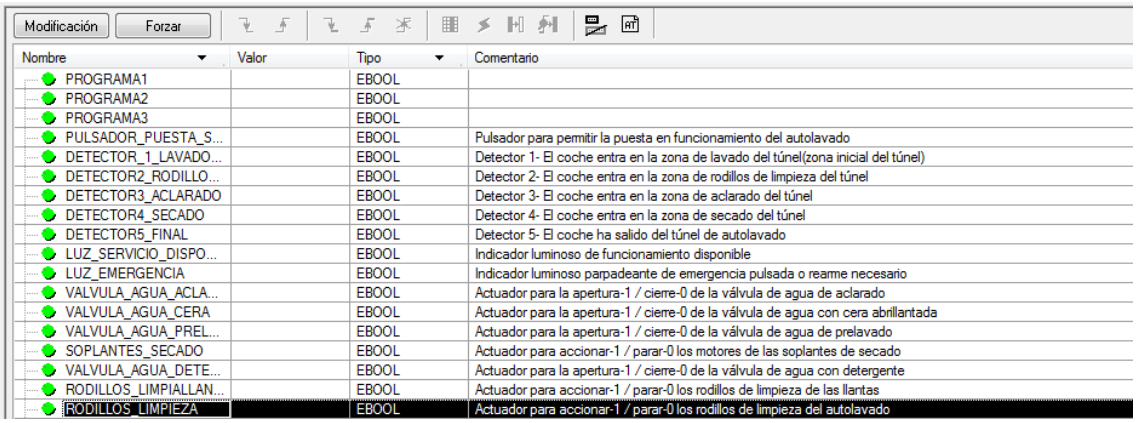


Figura 72: Creación de tabla de animación

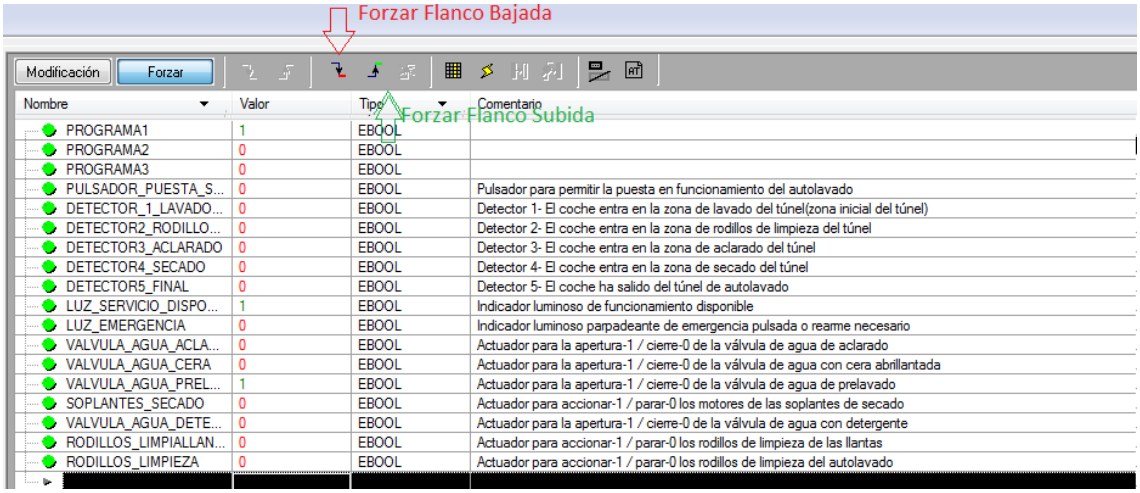
A continuación introducimos las variables que deseamos forzar o visualizar su resultado:



Nombre	Valor	Tipo	Comentario
PROGRAMA1		EBOOL	
PROGRAMA2		EBOOL	
PROGRAMA3		EBOOL	
PULSADOR_PUESTA_S...		EBOOL	Pulsador para permitir la puesta en funcionamiento del autolavado
DETECTOR_1_LAVADO...		EBOOL	Detector 1- El coche entra en la zona de lavado del túnel(zona inicial del túnel)
DETECTOR2_RODILLO...		EBOOL	Detector 2- El coche entra en la zona de rodillos de limpieza del túnel
DETECTOR3_ACLARADO		EBOOL	Detector 3- El coche entra en la zona de aclarado del túnel
DETECTOR4_SECADO		EBOOL	Detector 4- El coche entra en la zona de secado del túnel
DETECTOR5_FINAL		EBOOL	Detector 5- El coche ha salido del túnel de autolavado
LUZ_SERVICIO_DISPO...		EBOOL	Indicador luminoso de funcionamiento disponible
LUZ_EMERGENCIA		EBOOL	Indicador luminoso parpadeante de emergencia pulsada o rearme necesario
VALVULA_AGUA_ACLA...		EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua de aclarado
VALVULA_AGUA_CERA		EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua con cera abrillantada
VALVULA_AGUA_PREL...		EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua de prelavado
SOPLANTES_SECADO		EBOOL	Actuador para accionar-1 / parar-0 los motores de las soplantes de secado
VALVULA_AGUA_DETE...		EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua con detergente
RODILLOS_LIMPIALLAN...		EBOOL	Actuador para accionar-1 / parar-0 los rodillos de limpieza de las llantas
RODILLOS_LIMPIEZA		EBOOL	Actuador para accionar-1 / parar-0 los rodillos de limpieza del autolavado

Figura 73: Variables de la tabla de animación

Una vez compilado el proyecto, podremos comenzar a forzar variables y ver el estado de cada una de ellas:



Nombre	Valor	Tipo	Comentario
PROGRAMA1	1	EBOOL	
PROGRAMA2	0	EBOOL	
PROGRAMA3	0	EBOOL	
PULSADOR_PUESTA_S...	0	EBOOL	Pulsador para permitir la puesta en funcionamiento del autolavado
DETECTOR_1_LAVADO...	0	EBOOL	Detector 1- El coche entra en la zona de lavado del túnel(zona inicial del túnel)
DETECTOR2_RODILLO...	0	EBOOL	Detector 2- El coche entra en la zona de rodillos de limpieza del túnel
DETECTOR3_ACLARADO	0	EBOOL	Detector 3- El coche entra en la zona de aclarado del túnel
DETECTOR4_SECADO	0	EBOOL	Detector 4- El coche entra en la zona de secado del túnel
DETECTOR5_FINAL	0	EBOOL	Detector 5- El coche ha salido del túnel de autolavado
LUZ_SERVICIO_DISPO...	1	EBOOL	Indicador luminoso de funcionamiento disponible
LUZ_EMERGENCIA	0	EBOOL	Indicador luminoso parpadeante de emergencia pulsada o rearme necesario
VALVULA_AGUA_ACLA...	0	EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua de aclarado
VALVULA_AGUA_CERA	0	EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua con cera abrillantada
VALVULA_AGUA_PREL...	1	EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua de prelavado
SOPLANTES_SECADO	0	EBOOL	Actuador para accionar-1 / parar-0 los motores de las soplantes de secado
VALVULA_AGUA_DETE...	0	EBOOL	Actuador para la apertura-1 / cierre-0 de la válvula de agua con detergente
RODILLOS_LIMPIALLAN...	0	EBOOL	Actuador para accionar-1 / parar-0 los rodillos de limpieza de las llantas
RODILLOS_LIMPIEZA	0	EBOOL	Actuador para accionar-1 / parar-0 los rodillos de limpieza del autolavado

Figura 74: Señales que se pueden forzar

Es posible visualizar un valor utilizando otro formato de visualización: binario, decimal, hexadecimal, ASCII.
Si no forzamos los bits no podemos avanzar en el proceso, puesto que será necesario. En este caso funcionará tal y como se ha programado y en función de la opción que elija el usuario.

6. CONCLUSIONES Y APLICACIONES FUTURAS

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

6 CONCLUSIONES Y APLICACIONES FUTURAS

6.1 Conclusiones

La realización de este proyecto de fin de grado ha servido como complemento a la asignatura de 2º que es Automatización Industrial I y la optativa de 4º que es Automatización Industrial II. El trabajo realizado permite una visión global de la programación que hay detrás de una instalación de este tipo. Por otro lado el trabajo con una pantalla de visualización del proceso con elementos industriales y la familiarización con sus componentes dotan al alumno de una educación más práctica para dar un toque final a su carrera.

En cuanto al resultado del proyecto, se considera que los objetivos están cumplidos, al menos en la medida de lo posible:

- La programación del autómata en Unity Pro es más versátil que con el software previo de Schneider PL7 Junior.
- Creación de un manual de la herramienta del simulador. En base a esto, se ha hecho un recorrido por la simulación completa de un ejemplo, para analizar paso a paso como se puede aprovechar el simulador del software y cuáles son los puntos más fuerte del software y como configurar de manera correcta hacia el objetivo que propusimos.
- Cuenta con un documento de aplicación a la docencia, en donde al alumno, se le da una serie de pautas, para que pueda programar la estación creada utilizando el lenguaje que estime oportuno.
- La aportación del simulador a la Universidad, es permitir complementar las horas reglamentarias de realización de prácticas en los laboratorios de automatización de los alumnos, ofreciendo un entorno para poder trabajar en casa con las mismas características que el entorno de laboratorio, de esta manera lo que se consigue es optimizar en la enseñanza y ofrecer herramientas de auto aprendizaje y refuerzo de lo aprendido para el alumno.
- En este TFG se ha puesto especial interés en explicar el desarrollo del proyecto paso a paso de forma que el alumno vea con claridad que el software ofrece un manejo sencillo y una adaptación al entorno adecuada.

Creo que los objetivos del proyecto se han cumplido y se ha explicado la gran versatilidad que Unity Pro ofrece al programador.

6.2 Aplicaciones futuras

Hoy en día, tenemos una gran disponibilidad y capacidad de acceso a las nuevas tecnologías, de tal manera que el desarrollo de tecnologías Web, adquisición de datos, control y supervisión de procesos industriales están al alcance de cualquiera, por lo que te permite un auto-aprendizaje que con ayuda de la docencia recibida en la Universidad puedes trabajar y poner en práctica todos los conocimientos sin tener que asistir a los laboratorios de la misma.

Los objetivos que se propone nuestra Escuela Politécnica Superior es proporcionar una buena formación a sus alumnos, para ello debe disponer de buenas instalaciones, aparatología y laboratorios para su uso. Dentro de la formación, la práctica es uno de los factores más importantes dentro de cada asignatura. El alumno a través de los conceptos teóricos adquirirá conocimientos adecuados que a la hora de la realización de la práctica tiene que hacer uso de ellos. Los simuladores docentes para la realización de este proyecto han sido clave.

Un gran objetivo sería, poder conseguir un simulador virtual general que se pudiese utilizar con todas las maquetas en comparación con este caso donde se realiza uno particular para cada maqueta.

Por parte de la Universidad Carlos III, sería habilitar más recursos informáticos para el acceso a los simuladores, y poder regular de manera más adecuada el uso de licencias del propio software.

Por último, podríamos seguir haciendo en la Universidad una mejora del entorno de simulación de cara a su aplicación docente para que los futuros alumnos trabajen con una mayor autonomía, responsabilidad, aprendizaje, estudio continuado, aprendizaje colaborativo, en definitiva, mejores recursos.

7. BIBLIOGRAFÍA

Simulación de la Automatización de Procesos con Unity Pro

Guillermo Calvo Guadaño
U. Carlos III de Madrid
Dpto. Sistemas y Automática

7 BIBLIOGRAFÍA

[1] Diccionario de la Real Academia Española. <<Definición de automatización>> [En línea]. Available: http://lema.rae.es/drae/?val=automatizacion.

[2] <<Automatización>> [En línea]. Available: https://es.wikipedia.org/wiki/Automatizacion_industrial [Último acceso: Mayo 2013]

[3] Blanco, Dolores. Práctica 1de autómatas programables en el entorno de programación con Unity Pro, 2012.

[4] <<Simulator>> [En línea]. Available: https://es.wikipedia.org/wiki/Simulador [Último acceso: Mayo 2013]

[5] Sánchez Benítez, Gema. <<Las estrategias de aprendizaje a través de componente lúdico>> http://marcoele.com/descargas/11/sanchez-estrategias-ludico.pdf [Último acceso: Junio 2013]

[6] Ruiz Gutiérrez, José Manuel. La simulación como Instrumento de aprendizaje. http://mami.uclm.es/jmruiz/materiales/Documentos/simulacion.PDF

[7] Estandarización en la programación de control industrial. IEC 61131-3 <<Un recurso de programación estándar>>

[8] <<Scada>> [En línea]. Available: https://es.wikipedia.org/wiki/SCADA [Último acceso: Mayo 2013]

[9] BRAVO, Ignacio. Entornos de automatizaciones industriales [En línea] [Universidad de Alcalá, España]. Available: http://193.146.57.132/depeca/repositorio/asignaturas/201608/T3_VIJEO_2_comunic_PLC.pdf [Último acceso: Mayo de 2013]

[10] Proyectos de fin de Carrera. Universidad de Cádiz <<Lenguaje ST>> [En línea]. Available: http://rodin.uca.es:8081/xmlui/bitstream/handle/10498/12123/3514936x.pdf?sequence=1 [Último acceso: Mayo de 2013]

[11] Sistemas de Control. <<Listados de Intrucciones (IL)>> [En línea]. Available: http://inf7fj0.blogspot.com.es/p/plc.html [Último acceso: Mayo de 2013]

[12] <<Lenguaje de Contactos (LD) >> [En línea]. Available: http://www.infoplc.net/files/descargas/rockwell/infoPLC_net_Ejemplo_Programacion_Control_Logix.pdf [Último acceso: Junio de 2013]

[13] <<Esquema básico de funciones (FBD) >> [En línea]. Available: http://www.ingeborda.com.ar/biblioteca/Biblioteca%20Internet/Articulos%20Tecnicos%20de%20Consulta/Redes%20de%20Datos/PLC/T2_lenguajes_programacion_V1.pdf [Último acceso: Junio de 2013]